

IBM

Store Integrator

Programming Guide (Draft)

V1.0

Note: Before using this information and the product it supports, be sure to read the general information under "Notices," on page 216.

First Edition (July 2004)

This edition applies to Version 1 of the licensed program Store Integrator (program number 5639-P49) and to all subsequent releases and modifications until otherwise indicated in new editions.

You can order IBM publications through your IBM representative or the IBM branch office that serves your locality. Requests for copies of this publication and for technical information about IBM products should be made to your IBM Marketing Representative. Publications are not stocked at the address that is given below.

This publication is available on the IBM Retail Store Solutions Web site.

1. Go to www.ibm.com/solutions/retail/store.
2. Click Support.
3. Click Publications.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

Department CNPA
Design and Information Development
IBM Corporation
PO Box 12195
Research Triangle Park, North Carolina, 27709 U.S.A.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2004. All rights reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

FIGURES	X
TABLES.....	XI
ABOUT THIS GUIDE.....	XIII
WHO SHOULD READ THIS GUIDE.....	XIII
USING THIS GUIDE	XIII
CHAPTER 1. OVERVIEW	1
AEF OBJECTIVES	1
AEF ARCHITECTURE OVERVIEW	1
Real vs. Virtual Sessions	3
AEF Programming Tasks.....	4
SYSTEMS MANAGEMENT OBJECTIVES	6
SYSTEMS MANAGEMENT OVERVIEW	6
Systems Management Disciplines.....	6
Java ManagementExtensions (JMX).....	7
MBeans.....	7
Mid-Level Manager	7
Management Model	8
Systems Management Programming Tasks	10
CHAPTER 2. UNDERSTANDING THE ARCHITECTURE.....	11
INFRASTRUCTURE.....	11
AEFSession	11
AEFSessionFactory	12
AEFSessionPool.....	13
SessionServer	13
LoadBalancer.....	14
AEFBase.....	14
AEFException	14
Component Configurations.....	15
Configuration Properties.....	17
AUTOMATION PROVIDER	19
Error Handling Modes.....	19
Identifier Arguments.....	20
Info Objects.....	22
Configuration Properties.....	23
POSAutomationProvider.....	24
Operator.....	24
Customer	25
Transaction	25
SalesTransaction	25
LineItem	25
Item	26
Coupon	26
Discount.....	26
Points	26
Tender.....	27
CreditTender.....	27
Using Remote References	27
AUTOMATION API “UNDER THE COVERS”	27
Action Classes	27
Property Conditions	29

Bad Conditions	33
ConditionLock	33
ObjectDetectorLock	34
AEF ERROR HANDLING	34
Determining the Error Key	35
Error Helper Classes	36
DATA PROVIDER.....	44
Properties.....	44
Event and Listener Types	47
Extending POSAppEvents.....	52
Listener Considerations	53
SYSTEMS MANAGEMENT	55
Components.....	55
Agent Discovery and Health Checking	56
Notification Control	58
Logging Configuration and Forwarding	59
Inventory	61
Store Integrator Viewer.....	61
CHAPTER 3. PROGRAMMING WITH THE AEF	63
GETTING STARTED	63
Task: Setting up the Development Environment	63
Task: Using the Quick Start Sample.....	65
AEF	67
Infrastructure.....	67
Data Provider	71
Automation Provider	94
SYSTEM MANAGEMENT	107
Task: Writing and Registering Your Own MBeans	107
Task: Interfacing with the MgmtNotificationControlMBean:.....	112
CODING GUIDELINES	113
Task: Using Error Logging	113
Task: Writing an Error Helper	117
Task: Handling Exceptions	125
BUILDING, DEBUGGING AND PACKAGING.....	133
Task: Using Properties File Configuration	133
Task: Packaging Your Code	137
Task: Using the 4690 Debugger for POS Applications under CSS.....	138
CHAPTER 4. AEF ENABLING A POS APPLICATION.....	140
AEF CONFIGURATION SETTINGS	140
Default Operator ID/Passwords.....	140
Automatic Manager Override Number	140
Tender Mappings.....	141
POS APPLICATION HOOKS	141
POS Application Events.....	141
ENABLING USER EXTENSIONS AND ADDITIONAL FEATURES.....	144
Sending Additional POS Data to the AEF	144
Overriding or Extending the Automation Provider API	144
APPENDIX A. ERROR CODES	146
AEFCONST ERROR CODES	146
AEFCONST EXTENDED ERROR CODES	153
APPENDIX B. AEF PROPERTY FILES.....	158
4690 FILENAME RESTRICTION.....	158
AUTOMATION BUNDLE.....	158
Automation.properties.....	158

Appautomation.properties (ACE).....	158
Appautomation.properties (GSA)	159
Appautomation.properties (SA)	159
CLASS BUNDLE	160
Classes.properties	160
Appclass.properties (ACE)	163
Appclass.properties (GSA)	166
Appclass.properties (SA)	169
CONFIGURATION BUNDLE.....	171
Config.properties	171
Appconfig.properties (ACE)	176
Appconfig.properties (GSA).....	177
Appconfig.properties (SA).....	177
ERROR BUNDLE	177
Error.properties	178
Apperror.properties (ACE)	178
Apperror.properties (GSA).....	211
Apperror.properties (SA).....	234
FUNCTION CODE BUNDLE	256
Fcode.properties	256
Appfcode.properties (ACE).....	256
Appfcode.properties (GSA).....	257
Appfcode.properties (SA)	258
INTERNALIZATION BUNDLE	259
I18ntext.properties	259
Appi18ntext.properties (ACE)	259
Appi18ntext.properties (GSA).....	259
Appi18ntext.properties (SA).....	260
KEY SEQUENCE BUNDLE.....	260
Sequence.properties.....	260
Appseq.properties (ACE)	260
Appseq.properties (GSA).....	261
Appseq.properties (SA)	262
LOGON BUNDLE	263
Applogon.properties (ACE).....	263
Applogon.properties (GSA).....	263
Applogon.properties (SA)	263
SESSION BUNDLE	264
Session.properties	264
Appsession.properties (ACE)	264
Appsession.properties (GSA)	265
Appsession.properties (SA)	265
STATE BUNDLE.....	265
States.properties	265
Appstate.properties (ACE).....	265
Appstate.properties (GSA).....	266
Appstate.properties (SA)	268
SUBSTATE BUNDLE.....	269
Appsubstates.properties (ACE)	269
Appsubstates.properties (GSA).....	276
Appsubstates.properties (SA).....	276
TENDER MAP BUNDLE	278
Tendermap.properties	278
Apptendermap.properties (ACE)	278
Apptendermap.properties (GSA)	279
Apptendermap.properties (SA).....	279

APPENDIX C. DATA PROVIDER PROPERTIES.....	280
COUPONPROPERTIES	280
CUSTOMERPROPERTIES	282
DISCOUNTPROPERTIES	283
ITEMSALESPROPERTIES	284
LINEITEMPROPERTIES	287
OPERATORAUTHORIZATIONPROPERTIES	289
Authorization Properties (Common)	289
Authorization Properties (GSA)	289
Authorization Properties (SA & ACE)	290
OPTIONSPROPERTIES.....	292
POINTSPROPERTIES	295
POSDEVICEPROPERTIES	296
STOREOPTIONSPROPERTIES	300
TENDERPROPERTIES.....	304
TERMINALOPTIONSPROPERTIES.....	305
TRANSACTIONSTATUSPROPERTIES	308
TRANSACTIONTOTALSPROPERTIES	312
WORKSTATIONSTATUSPROPERTIES	314
APPENDIX D. POS APPLICATION XML EVENTS.....	316
OPERATOR EVENTS.....	316
signOn.....	316
signOff.....	318
lock.....	318
unlock.....	318
customer	318
POINTS EVENT	321
points	321
TRANSACTIONTOTALS EVENT.....	322
transactionTotals	322
TRANSACTIONSTATUS EVENTS	323
transactionStart.....	323
transactionVoid	326
transactionSuspended	328
transactionEnd.....	330
transactionUpdate.....	332
transactionTaxChange.....	332
ITEMSALES EVENT	333
itemEvent	333
COUPON EVENT	336
coupon	336
DISCOUNT EVENTS	338
transactionDiscount	338
lineItemDiscount	339
TENDER EVENT	341
Tender.....	341
CASHRECEIPT EVENT.....	343
cashReceipt	343
SCALE EVENT	343
scale.....	343
REPORT EVENT	343
report.....	343
DATAEVENT	344
dataEvent.....	344
WORKSTATIONSTATUS EVENT	345

workstationStatus 345

OPTIONSEVENT 345

Options..... 345

APPENDIX E. SALOGONACTIONIMPL SOURCE EXAMPLE 358

APPENDIX. NOTICES..... 368

TRADEMARKS..... 370

Figures

FIGURE 1 AEFSESSION ARCHITECTURE	2
FIGURE 2 4690 TERMINAL SESSION	3
FIGURE 3 4690 CSS SESSIONS	4
FIGURE 4 MID-LEVEL MANAGER	8
FIGURE 5 STORE INTEGRATION FRAMEWORK MANAGEMENT MODEL.....	10
FIGURE 6 AEFSESSION COMPONENTS	12
FIGURE 7 GET AVAILABLE SESSION SEQUENCE	13
FIGURE 8 TYPICAL STORE CONFIGURATION	15
FIGURE 9 EXECUTE A TRANSACTION SEQUENCE	19
FIGURE 10 IDENTIFIER INTERFACE HIERARCHY	22
FIGURE 11 INFO OBJECT INTERFACES.....	23
FIGURE 12 ACTION CLASSES	29
FIGURE 13 DATA PROVIDER PROPERTY HIERARCHY	46
FIGURE 14 POSDATAProvider EVENT HIERARCHY	47
FIGURE 15 MANAGEMENT INFRASTRUCTURE.....	56
FIGURE 16 NOTIFICATION CONTROL MECHANISM	59
FIGURE 17 STORE INTEGRATOR VIEWER	62
FIGURE 18 SAMPLEGUI INITIAL PROGRESS SCREEN	66
FIGURE 19 SAMPLEGUI MAIN SCREEN	66
FIGURE 20 DISPLAYING A DATA PROVIDER PROPERTY VALUE	72
FIGURE 21 AEFPROPERTYCHANGELISTENER EXAMPLE.....	74
FIGURE 22 SAMPLEGUI ACTION CLASSES	95
FIGURE 23 AEF ENABLED APPLICATION.....	143

Tables

TABLE 1 AEFBASE CONFIGURATION PROPERTIES	17
TABLE 2 AEFSESSION CONFIGURATION PROPERTIES.....	18
TABLE 3 AUTOMATION PROVIDER CONFIGURATION PROPERTIES	24
TABLE 4 PROPERTY CONDITION CLASSES	33
TABLE 5 ERROR HELPERS.....	43
TABLE 6 DATA PROVIDER PROPERTY CATEGORIES	45
TABLE 7 DATA PROVIDER EVENT LISTENER PROXIES	54
TABLE 8 LOGGING MBEANS.....	60
TABLE 9 REMOTELOGGINGCTRLMBEAN LOG LEVELS	61
TABLE 10 DEVELOPMENT MACHINE DIRECTORY HIERARCHY	64
TABLE 11 SA USER EXIT EVENT NUMBERS	77
TABLE 12 LOGGING LEVELS.....	114
TABLE 13 LOG LEVELS WITH SAMPLE LOG ENTRIES	114
TABLE 14 USER JAR FILE NAMES	133
TABLE 15 JAR COMMANDS.....	134
TABLE 16 AEF ENABLED APPLICATIONS	141
TABLE 17 AEFEXCEPTION ERROR CODES.....	152
TABLE 18 AEFEXCEPTION EXTENDED ERROR CODES	157
TABLE 19 DATA PROVIDER COUPON PROPERTIES	282
TABLE 20 DATA PROVIDER CUSTOMER PROPERTIES.....	282
TABLE 21 DATA PROVIDER DISCOUNT PROPERTIES	283
TABLE 22 DATA PROVIDER ITEMSALES PROPERTIES	286
TABLE 23 DATA PROVIDER LINEITEM PROPERTIES	288
TABLE 24 DATA PROVIDER AUTHORIZATION PROPERTIES (COMMON)	289
TABLE 25 DATA PROVIDER AUTHORIZATION PROPERTIES (GSA).....	290
TABLE 26 DATA PROVIDER AUTHORIZATION PROPERTIES (SA & ACE).....	291
TABLE 27 DATA PROVIDER OPTIONS PROPERTIES.....	294
TABLE 28 DATA PROVIDER POINTS PROPERTIES	295
TABLE 29 DATA PROVIDER POSDEVICE PROPERTIES	299
TABLE 30 DATA PROVIDER STOREOPTIONS PROPERTIES	303
TABLE 31 DATA PROVIDER TENDER PROPERTIES.....	304
TABLE 32 DATA PROVIDER TERMINALOPTIONS PROPERTIES.....	307
TABLE 33 DATA PROVIDER TRANSACTIONSTATUS PROPERTIES	311
TABLE 34 DATA PROVIDER TRANSACTIONTOTALS PROPERTIES	313
TABLE 35 DATA PROVIDER WORKSTATIONSTATUS PROPERTIES	315
TABLE 36 <SIGNON> EVENT	316
TABLE 37 OPERATOR AUTHORIZATION ATTRIBUTES	317
TABLE 38 <SIGNOFF> EVENT.....	318
TABLE 39 <LOCK> EVENT	318
TABLE 40 <UNLOCK> EVENT	318
TABLE 41 <CUSTOMER> EVENT	320
TABLE 42 <POINTS> EVENT	321
TABLE 43 <TRANSACTIONTOTALS> EVENT.....	323
TABLE 44 <TRANSACTIONSTART> EVENT.....	325
TABLE 45 <TRANSACTIONVOID> EVENT.....	327
TABLE 46 <TRANSACTIONSUSPENDED> EVENT	329
TABLE 47 <TRANSACTIONEND> EVENT	331
TABLE 48 <TRANSACTIONUPDATE> EVENT	332
TABLE 49 <TRANSACTIONTAXCHANGE> EVENT.....	332
TABLE 50 <ITEMEVENT> EVENT	335
TABLE 51 <COUPON> EVENT.....	338

TABLE 52 <TRANSACTIONDISCOUNT> EVENT.....	339
TABLE 53 <LINEITEMDISCOUNT> EVENT.....	340
TABLE 54 <TENDER> EVENT	342
TABLE 55 <CASHRECEIPT> EVENT.....	343
TABLE 56 <SCALE> EVENT.....	343
TABLE 57 <REPORT> EVENT	344
TABLE 58 <DATAEVENT> EVENT	344
TABLE 59 <DATAEVENT> VALUES.....	345
TABLE 60 <WORKSTATIONSTATUS> EVENT.....	345
TABLE 61 <OPTIONS> EVENT	357

About this guide

This guide shows you how to integrate or extend point of sale applications using the IBM Application Extension Facility (AEF).

Who should read this guide

This guide is intended for programmers interested in using the AEF to develop extensions to point of sale applications, or in using the AEF to integrate with point of sale logic. Those responsible for planning the installation and use of AEF, using the Store Integration Framework's system management facilities, and interested in customizing the system through the configuration process should consult the *Store integrator User's Guide* (IBM Form Number SC30-4085). If planning to customize the Store Integrator Graphical User Interface (SI GUI), then refer to *IBM 4690 Store Integrator Graphical User Interface Programming Guide*. A working knowledge of the point-of-sale (POS) application currently used by the customer, such as the IBM 4690 General Sales Application (GSA), Supermarket Application (SA), or SurePOS Application Client/Server (ACE), the programming language used by that application (C-BASIC or C++), XML and the Java programming language, and of the IBM 4690 Operating System, is recommended for those working with this material.

Using this guide

The documentation for programming the AEF consists of three parts:

1. The Programming Guide (which you are currently reading).
2. A Javadoc version of the AEF API interface. Javadoc is a standard HTML format for documenting a Java API. Although the Javadoc for the AEF can stand on its own as an API reference, the Programming Guide also contains hyperlinks into the [AEF Javadoc](#).
3. Code samples. The Programming Guide makes use of code samples to illustrate AEF API usage. Rather than include entire code modules within the Programming Guide, hyperlinks to the code samples are used instead.

The Programming Guide, javadoc, and code samples are all included in the Zip archive from which this Programming Guide was extracted. In order for the hyperlinks to work properly, the Zip files must be extracted into a directory called "si" on the root of your file system. The relative directory structure within the Zip archive must be preserved when extracting into the "si" directory.

Chapter 1 of this guide contains a high level overview of the objectives and architecture of the AEF. Chapter 2 presents a more detailed discussion of the AEF architecture. Chapter 3 gives specific explanations and examples of programming with the AEF API. Included at the beginning of Chapter 3 is a "quick start" guide for getting a working example of client code up and running.

Chapter 1. Overview

The IBM Application Extension Facility (AEF) is a component of the IBM Store Integrator. The AEF is a Java application programming interface (API) which allows client code to observe and interact with working point of sale sessions. In order to understand how to use the AEF API, it is necessary to gain some understanding of its objectives and architecture.

AEF Objectives

The objectives of the AEF are:

- Provide programmatic access to point of sale (POS) functions via a Java API.
 - Use of the API should not require extensive retail domain knowledge.
 - The API should allow remote access.
 - The API should provide a level of abstraction from the POS application such that client code may be somewhat independent from the POS application.
- Provide for integrating POS logic into other applications. Rather than attempt to duplicate complicated point of sale application logic, the objective is to provide access to existing application logic. This is done by allowing the POS application to run in a “virtual” register session. An example of integrating with POS logic is a web based home shopping application. The home shopping application is implemented as a servlet which utilizes virtual POS sessions from the customer’s home store. Since the servlet is using the same POS application logic, the customer using the browser based home shopping solution will be provided the same pricing information as an actual register in the store would provide.
- Provide for extending the POS application via Java. The same Java API which applies to virtual POS register sessions can be applied to real POS registers. This allows register extensions to be written in Java. An example of this type of extension is integrating an RFID reader with a POS register. The AEF API may be used to cause a new operator key sequence to trigger the RFID reader to scan the contents of a basket. The AEF API may then be used to inject the items into the transaction on the POS register.
- Provide systems management functions consisting of monitoring and control of real and virtual POS sessions and their environments.

AEF Architecture Overview

One of the AEF objectives is to provide a standard Java interface to multiple POS applications. For this release of Store Integrator, the following applications support the AEF API:

IBM 4680-4690 General Sales Application (GSA)
IBM 4680-4690 Supermarket Application (SA)
IBM SurePOS Application Client/Server Environment for 4690 OS (ACE)

Within the basic AEF architecture, the POS terminal sales application is enhanced with a set of hooks which generate XML events as illustrated in the following figure. These events indicate the state of the POS application. Events are generated for POS actions such as operator logon/logoff, start or end of transaction, item sale, tender, etc.

The AEF utilizes these events to maintain a Java object model representing the state of the POS application. Each instance of the POS terminal sales application is wrapped by an instance of [AEFSession](#). The session object and the Java objects contained within the session together form the Java AEF API which may be used by clients to interact with the POS logic. The AEF API can also drive the POS terminal sales application by generating device input such as keyboard or scanner input.

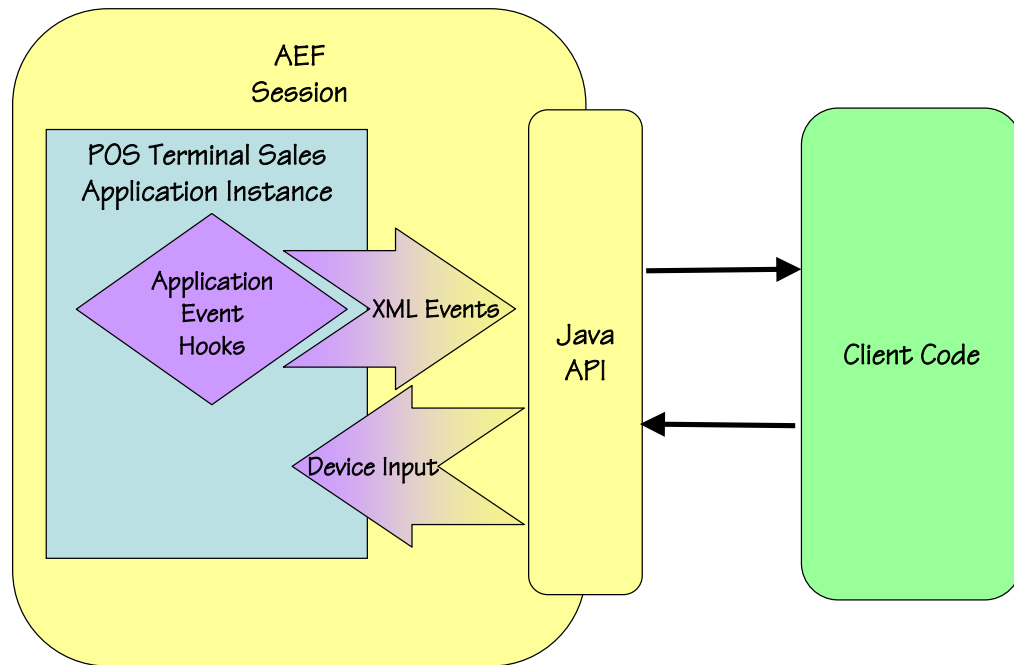


Figure 1 AEFSession Architecture

Real vs. Virtual Sessions

AEF sessions may be configured to run on a physical 4690 register, or on a virtual register within the Client Session Server (CSS) environment.

To run on a physical 4690 terminal, the terminal must be configured to run the terminal sales application as well as the AEF. In this configuration, the client Java code typically runs inside the same terminal Java virtual machine (JVM) as does the AEF. The following figure illustrates the AEF running in a physical 4690 terminal, with the client code coresident within the same JVM. Since the SI GUI is a client of the AEF API, it is an example of this configuration when running on a 4690 register.

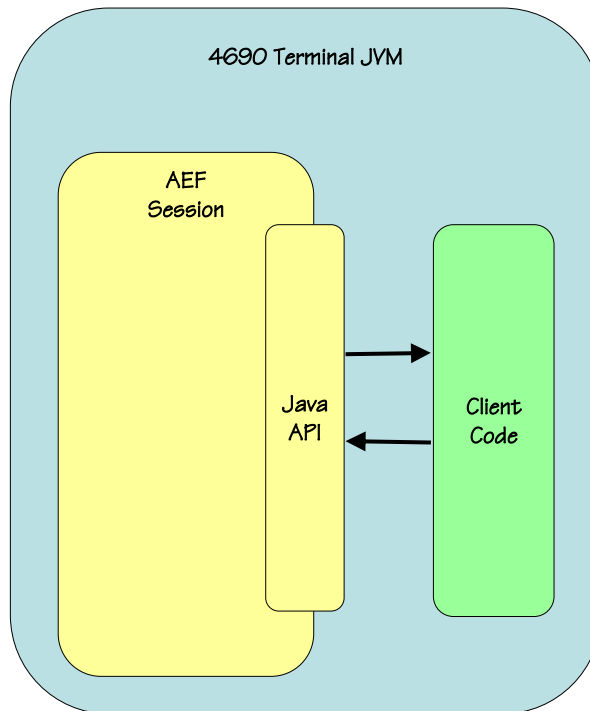


Figure 2 4690 Terminal Session

Virtual AEF sessions may be configured to run on 4690 controllers. In this configuration, the same terminal sales program is run on the 4690 controller as the one which runs in the physical registers, with the exception that all IO devices are emulated. There is a separate Java AEFSession object wrapping each instance of the terminal sales application. The client (which is typically remote) connects with one or more virtual sessions within the Client Session Server (CSS) JVM. The following figure illustrates CSS with multiple virtual sessions.

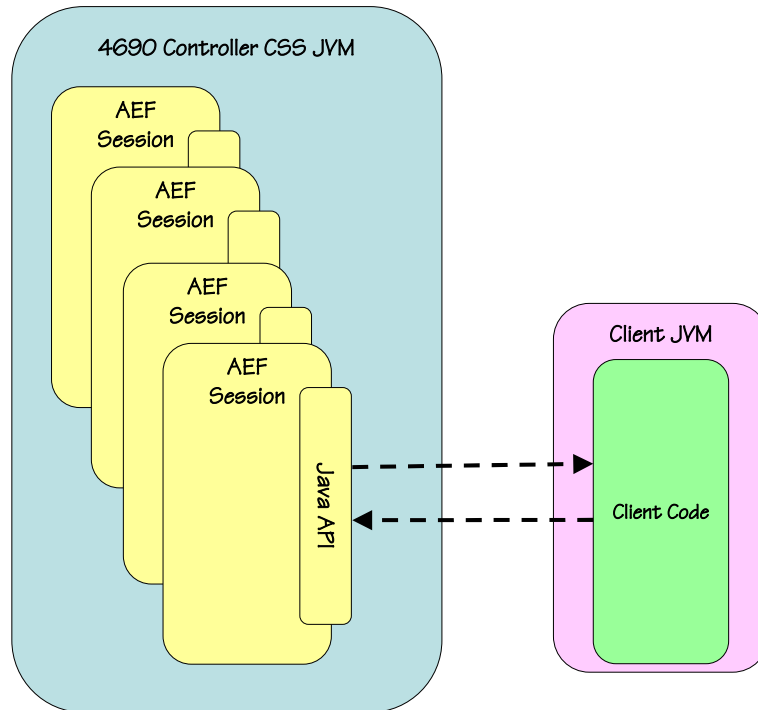


Figure 3 4690 CSS Sessions

AEF Programming Tasks

The following is a summary of Store Integrator AEF programming tasks:

- Setting up the Development Environment
- Using the QuickStart Example
- Getting a SessionServer
- Connecting to a Specific SessionServer
- Getting a Specific Session
- Getting a Session Without a Ready Wait
- Retrieving a Data Provider Property Value
- Subscribing to a Data Provider Event
- Adding Additional Data to an Event
- Sending a New XML Event from the POS Application
- Using an AEFSession
- Extending Existing Automation Provider APIs
- Extending the Automation Provider API with a New Function
- Using Error Logging

- Writing an Error Helper
- Handling Exceptions
- Using Properties File Configuration
- Packaging User Code and Properties Files
- Using the 4690 Debugger for POS Applications under CSS

Specific examples of these tasks may be found within “Chapter 3. Programming with the AEF”.

Systems Management Objectives

The goals of the Store Integrator systems management architecture are:

- Provide a Single-Store view for management
- Use an open, standards based instrumentation model
- Maintain strict isolation between components and management tools, so that all management applications behave in a consistent manner

Systems Management Overview

Systems Management Disciplines

Systems management within the Store Integrator is responsible for managing the following set of disciplines:

Configuration	The architecture includes support for local and remote configuration of managed software and hardware components.
Inventory	The architecture includes support for local and remote inventory reporting of both software and hardware components. For Store Integrator software components, the inventory information typically includes product description, product manufacturer, and product version information (including maintenance levels).
Event Notification and Forwarding	Store Integrator hardware and software components may generate notification events which may be monitored by systems management applications. The events may include normal operational or error information. The Store Integrator architecture presents a multi-tiered framework for collecting and filtering events at the store level for access by an enterprise system management application.
Monitoring/Thresholding	The architecture includes support for the monitoring of various attributes of Store Integrator components. The number of available sessions within a CSS instance is an example of an attribute which may be monitored. The Store Integrator architecture includes support for dynamically creating policy based attribute monitors which generate event notifications. This relieves the enterprise management application from actively polling the attribute and creating monitors manually.
Remote Control	Components within the Store Integrator (including the AEF) provide various levels of remote control capabilities. For example, the AEF Session includes the capability to pass keyboard input into the session from a management

application.

Java ManagementExtensions (JMX)

The Store Integrator systems management architecture is based upon Java Management Extensions (or JMX). The Java Management Extensions (JMX) Specification describes a general management framework for any Java (or even non-Java) program. It is a multi-tier architecture that includes an instrumentation layer, MBeans; an agent layer, MBeanServer; and a management layer. Refer to the latest *Java Management Extensions (JMX) Specification* (<http://java.sun.com/products/JavaManagement/>) to obtain complete information about the JMX framework. JMX is used both internally and externally to implement the disciplines discussed above. The AEF includes a set of MBeans which include notification generation, control functions, and attribute querying and modification.

MBeans

MBeans are Java objects which expose a management interface for a managed resource. MBeans become useful for systems management when their interfaces are exported so that they may be accessed by remote management applications.

Mid-Level Manager

Given the need to isolate the management tools from the infrastructure and to treat the store as the point of management control, the Store Integrator utilizes a Mid-Level manager or Proxy approach. It introduces a new point of collection and control in the store that becomes the focus of all outside management actions, and is thus responsible for executing any and all management actions. Like an enterprise level management tool, it has an infrastructure that is unique to itself and interacts with its own agents to accomplish management functions. The only different between it and an enterprise manager is how it exposes its functionality. An enterprise manager might expose automation and user interfaces, whereas the Mid-Level manager will only expose a programming interface for interaction. Figure 4 illustrates what the Mid-Level Manager looks in the Store Integrator environment.

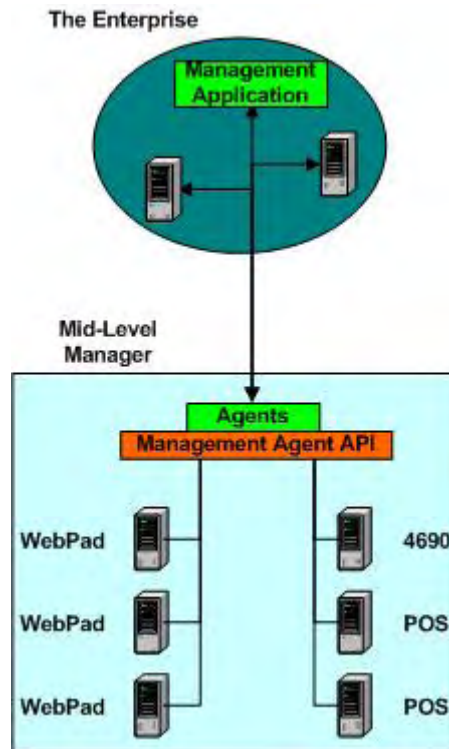


Figure 4 Mid-Level Manager

Figure 4 depicts how the Store Integration Framework is modeled using the Mid-Level Manager approach. All infrastructure deployed on the manageable devices is of a single implementation model, and is provided by the vendor of the device being managed, NOT by the management application vendor. The agents provided by management tool vendors are neatly contained above the mid-level manager (in this diagram, the mid-level manager is the Management Agent API) and have equal access to the system.

Management Model

The systems management model is broken down into well defined roles with well defined relationships between them. The intent is to make it very easy for developers of non-IBM software and users who want to build upon the software provided to know where in the model their component fits in, and what their responsibilities there are in building that component. All Systems Management code falls into one of the following four roles, and must adhere to the defined responsibilities for that role.

Management Sub-Agent (General Agent)

- Runs in every component that makes up the store environment.
- Defines manageable attributes and exposes them for interaction to the environment.
- Defines functional entry points as needed and exposes them for use within the environment.
- Defines and issues Notifications.

Management Agent API (Master Agent)

- Provides the implementation for communications between the Management Sub-Agents and Management Agent(s).
- Exposes a collective management interface representing all Management Sub-Agents for use by all Management Agents and Management Applications.
- Manages filters for and forwarding of notifications to Management Agent(s) and Management Application(s).
- Provides a point of implementation and control for store-wide policy.

Management Agent (Optional)

- A user of the Management Agent API for interacting with Management Sub-Agents
- Central point of contact and communication proxy for its corresponding management application
- Is the single presence of the Management vendors toolset in the Store Integration Framework environment
- Provided by Tivoli, ThinkVantage, or other provider

Management Application

- Located in the enterprise or the store
- Allows access (either automated or operator driven) to the Store Integrator systems management functions (MBeans)
- Provided by Tivoli, ThinkVantage, or others

The Store Integrator includes a basic browser based management application called the Store Integrator Viewer. Conceptually, the four elements of the model look like this:

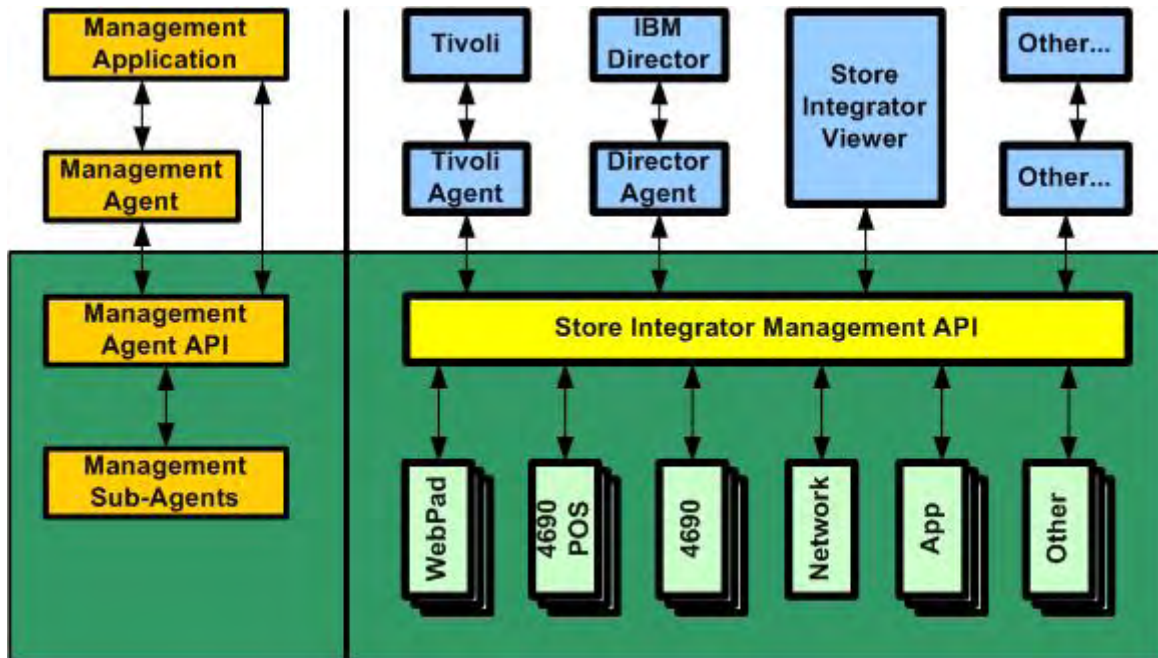


Figure 5 Store Integration Framework Management Model

Systems Management Programming Tasks

The following is a summary of the Store Integrator systems management programming tasks:

- Creating custom MBeans for systems management functions related to AEF extensions. Examples of systems management functions that developers may include in custom extensions are configuration, event notification, and control functions.
- Interfacing with systems management capabilities exposed via existing MBeans provided by the Store Integrator components.

Specific examples of these systems management tasks may be found within “Chapter 3. Programming with the AEF”.

Chapter 2. Understanding the Architecture

The AEF API is divided into the following categories:

Infrastructure	Includes classes which support the creating and remotely accessing sessions.
Automation Provider	A set of classes which provides an object model of the POS application and transactions. Includes synchronous operations such as starting transactions, adding items and tenders. Useful for actively manipulating the POS application.
Data Provider	Provides for passively observing events from the POS application. Includes event a listener interfaces.
Device Accessor	Allows the insertion of client device hooks which may observe, consume, or inject device IO.
Systems Management	Provides for remote monitoring and control of the AEF sessions and their environment. Client code may also participate in systems management functions.

Infrastructure

The following sections describe the primary infrastructure components of the AEF. These components are responsible for creating sessions, and providing remote access to them.

AEFSession

The AEFSession encapsulates access to a POS application running on a real register or a virtual register under the control of CSS. In the 4690 AEF environment, each terminal (real or virtual) will have an instance of AEFSession which supports both local and remote access. The AEFSession object is remotied via Java Remote Method Invocation (RMI). The AEFSession exposes three API sets as described below.

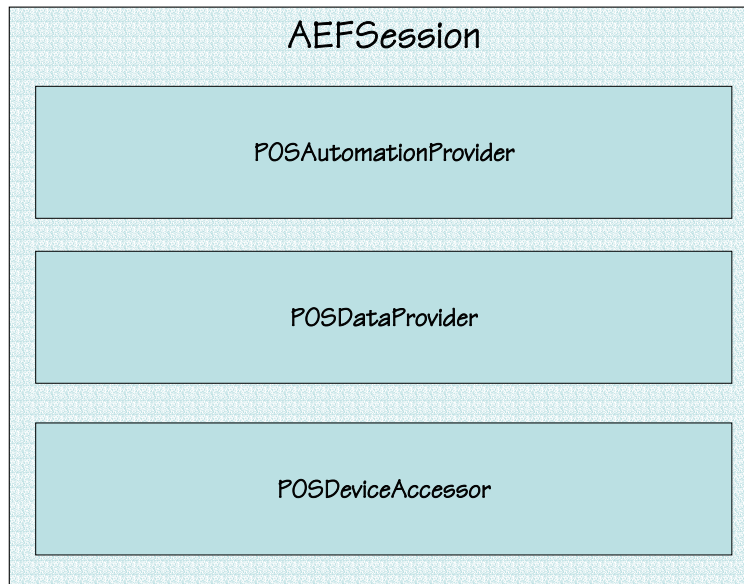


Figure 6 AEFSession Components

POSAutomationProvider

The POSAutomationProvider API may be used to actively drive the session. Examples include starting a transaction, and adding items and tenders to the transaction. This API is exposed via a Java object model that represents the transactions and line items for the register. The POSAutomationProvider API provides a certain level of POS application isolation so that common functions may be executed via the POSAutomationProvider API regardless of the underlying POS application.

POSDataProvider

The POSDataProvider API provides a mechanism for passively observing events occurring within the underlying POS application. This API supports both local and remote access.

POSDeviceAccessor

The Device Accessor API provides a JavaPOS interface for observing, modifying, injecting, and consuming POS IO device information. The API allows hooks to be installed between the POS application and the POS IO devices.

AEFSessionFactory

The [AEFSessionFactory](#) provides access to AEFSessions. On a real register, the AEFSessionFactory is able to return the single instance of AEFSession for that register. An AEFSessionFactory running in the CSS JVM is able to use CSS to start new virtual instances of the POS application. The AEFSessionFactory can return pre-existing virtual sessions, or create new ones as needed. Each factory maintains a pool of available sessions. Session factories can only exist on machines running 4690 OS. In a typical store, there will be an AEFSessionFactory instance in each real register, as well as an instance in each controller JVM running CSS.

Client code will not need to interact directly with the AEFSessionFactory. Instead, it will use the AEFBase class to gain access to a session. The AEFBase class will contact a SessionServer on the client's behalf. The SessionServer will then determine the appropriate AEFSessionFactory to handle the request.

Depending on AEF configuration options, the AEFSessionFactory can be exported via RMI. The factory *must* be exported if remote access to the sessions owned by the factory is required. By default, the factories on the 4690 controllers and 4690 terminals are exported. There is some overhead involved in exporting a factory, so if remote access to the factory's sessions is not desired, the AEF should be configured not to export the factory. For example, if remote access to terminal sessions are not required, it is recommended to configure the terminal factories as non-exported.

If a factory is configured as exported, it will periodically "beacon" to inform any SessionServers of its location.

AEFSessionPool

Each AEFSessionFactory contains a session pool (implementing the [AEFSessionPool](#) interface) which maintains a collection of sessions managed by that factory. The sessions may be acquired by the factory, and released back into the pool at a later time.

SessionServer

The [SessionServer](#) handles requests for sessions. If the request is for a specific terminal number, the SessionServer determines which factory can provide the requested session. If the request is for any available session, the SessionServer uses a load balancer to determine which factory should serve the request.

There may be multiple SessionServers configured within the store. Every SessionServer should know about all the exported factories in the store. Therefore, any SessionServer should be able to handle a request for a session, regardless of which factory ultimately provides the session.

The SessionServer is designed to listen for AEFSessionFactory within the same TCP/IP subnet.

The following sequence diagram illustrates how a session is fetched via a SessionServer.

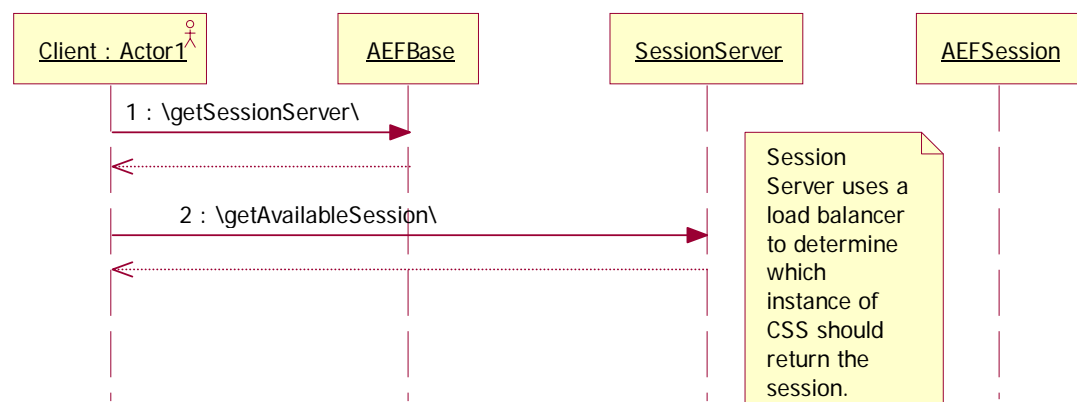


Figure 7 Get Available Session Sequence

LoadBalancer

Each SessionServer includes a [LoadBalancer](#) which helps determine which session factory should handle the next request for any available virtual session. Note that requests for real terminal sessions don't need load balancing. The default load balancing implementation is "round robin" allocation of session requests to each of the exported virtual session factories in the store.

The default load balancers have no knowledge of each other, so if there are multiple clients requesting available sessions from multiple session servers, the round robin behavior will only apply internally within each SessionServer, and not among all SessionServers.

The list of factories that the load balancer knows about is dynamic. Factories may come and go in the network. This may cause what appear to be "anomalies" in the round robin load balancing behavior, but is expected given the dynamic nature of the network.

If more sophisticated load balancing behavior is desired, the default load balancer implementation may be replaced via a factory mechanism.

AEFBase

The [AEFBase](#) is a class which is the "gateway" to the AEF. All JVMs which either run the AEF, or are clients of the AEF will create an instance of the AEFBase class. Depending upon the platform and configuration settings, the AEFBase may create and export instances of the SessionServer, the AEFSessionFactory, the DeviceServer (used for remote devices), the LoggerControl (used for setting logging levels), the DebugMemoryImpl (used to remotely query memory usage), and the RMI registry.

The AEFBase contains methods for requesting a specific AEFSession by terminal number, and for requesting any available virtual session. This is the recommended method for clients to gain access to an AEFSession. The AEFBase instance will contact a SessionServer which will utilize its LoadBalancer to determine which AEFSessionFactory should handle the request for the session.

AEFException

The AEF exception model defines a single exception class named [AEFException](#) <TODO: link> which is thrown to indicate errors. The particular error is indicated via a primary error code, and an extended error code.

Component Configurations

There are different configurations of the components discussed above which vary depending on the platform of execution. This is illustrated via the following diagram of a typical in-store configuration.

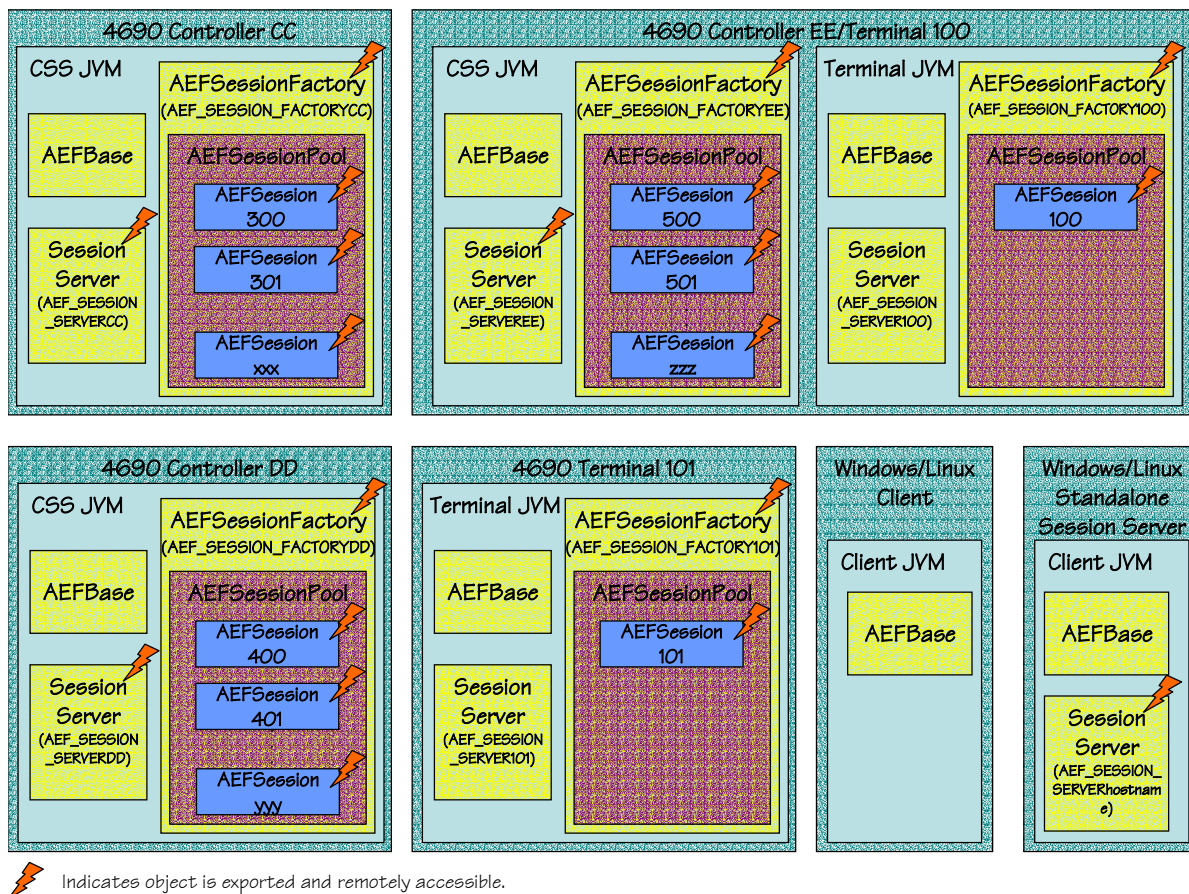


Figure 8 Typical Store Configuration

This figure illustrates a store which is using SIF to provide access to both virtual and real terminal sessions. CSS is configured to run on the 4690 controllers CC, DD, and on the controller/terminal EE. Each of these controllers contains an exported session server, and an exported session factory. Each of the session factories includes a session pool which manages the sessions. Note that as with real terminal numbers, the virtual terminal session numbers must be unique. They must not conflict with virtual terminal session numbers from other controllers, nor may they conflict with real terminal numbers in the store.

This example also demonstrates access to real terminal sessions. Terminal number 100 is the terminal portion of a 4690 controller/terminal, and terminal 101 is a standalone terminal. The session pools for real terminal sessions will always contain the single terminal session. The session factories for real terminals are exported by default, allowing remote access to the terminal session. However, it is more likely that access to the real terminal sessions will be local within the terminal JVM. For example, the SIF

GUI is a local client of the AEF (running in the same JVM as the AEF). Should no remote access to the real terminal sessions be required, the AEF may be configured not to export the terminal session factories. This would result in some network overhead reduction.

The figure above also shows a remote client which needs to gain access to a virtual session. This client could be on any platform supporting Java 1.4 or better, but is labeled as a Windows or Linux client in the figure. In this case the client creates an instance of AEFBase. The AEFBase instance may be used to gain a remote reference to a specific terminal session, or any available virtual terminal session.

The lower right rectangle in the figure represents a standalone session server. In certain store environments, there may be an advantage in having a session server independent from the 4690 controllers. The session servers maintain a list of all the factories in the store, and are capable of restoring factory connections if the factory machines are stopped and restarted. This means that a standalone session server could automatically restore references to the factories when the 4690 controllers and terminals are IPL'ed. The client can continue to use the standalone session server, even after the 4690 machines are IPL'ed.

Configuration Properties

The AEF may be configured via properties that effect various behavior of the AEF. The following sections will discuss some of the properties that effect AEFBase and AEFSessions.

AEFBase Configuration Properties

The following table documents configuration properties which impact the AEFBase. The default setting may be dependant on the platform of execution.

Behavior or Value	Default on 4690 Controller	Default on 4690 Terminal	Default on Other Platform	Configuration Property to Override
AEFBase creates session server	Yes	Yes	No	userconfig.properties create.server=true false
Session server is exported	Yes	No	N/A (Server not created by default)	userconfig.properties server.remote.access=true false
Session server ID	AEF_SESSION_SERVERxx where xx is the controller id	AEF_SESSION_SERVERyyy where yyy is the terminal number	AEF_SESSION_SERVERhostname where hostname is the TCP/IP hostname	userconfig.properties server.id=
AEFBase creates session factory	Yes	Yes	No (Factory not allowed)	userconfig.properties create.factory=true false
Factory is exported	Yes	Yes	N/A (Factory not allowed)	userconfig.properties factory.remote.access=true false
Factory ID	AEF_SESSION_FACTORYxx where xx is the controller id	AEF_SESSION_FACTORYyyy where yyy is the terminal number	N/A (Factory not allowed)	userconfig.properties factory.id=
AEFBase creates RMI registry	Yes	Yes	No	userconfig.properties create.rmi.registry=true false
RMI port	12099	11099	11099	userconfig.properties rmi.port=
AEFBase creates remotely accessible logger control	Yes	No	No	userconfig.properties logger.remote.access=true false
AEFBase creates device server for exporting devices to remote client	No	No	No	userconfig.properties create.device.server=true false
Device server port	-1	-1	-1	userconfig.properties device.server.port=

Table 1 AEFBase Configuration Properties

AEFSession Configuration Properties

The following table documents configuration properties which effect the behavior of AEFSessions.

Property	Notes	Default	Override
pos.sales.application	Set to the qualified path of the pos application to run.	SA: R::h0:/adx_ipgm/EAMTS10L.286 ACE: R::h0:/adx_ipgm/JSIFTS10.386 GSA: R::h0:/adx_ipgm/EALTS10L.286	usersession.properties
aeio.device.group	If a unique AEFIO device group is	Default	To override for all sessions belonging to a session role, put the property in appsession_sessionrol

	required, use the AEFIOCFG tool to create the device group, and set this property to the name of the new AEFIO device group. This should not be required except for special circumstances, such as running remote SIF GUI with remote devices.		e.properties. To override for a single session, put the property in appsession_XXX.properties where XXX is the terminal number.
start.pos.sales.application	<p>For virtual sessions, this property controls whether the pos application is automatically started when the session is created. If a client requests a virtual session which has not yet been created, this flag will determine if the pos app is started upon session creation. If set to false, the client may call <code>AEFSession.startApplication()</code> to cause the pos application to be launched. This is useful in situations where a client wants to observe or participate in the application startup. The client creates the session, installs any listeners and hooks, and then starts the application.</p> <p>For real terminal sessions, the client cannot directly control whether the pos application is started, but this property still has an effect. If this property is set to false, the IBM applications SA, ACE, and GSA will block early in their initialization processes waiting for the <code>AEFSession.startApplication()</code> to be called. Again, this is useful in situations where the client wishes to observe or participate in the pos application initialization.</p>	True	
session.extensions	Allows user code to be attached to an <code>AEFSession</code> . The code is meant to be run in the same JVM with the session. Set to a comma delimited list of classes which will be instantiated during session initialization. If the extension objects implement the <code>SessionExtension</code> interface, then the object's <code>initialize</code> method will be called.		

Table 2 AEFSession Configuration Properties

Automation Provider

The Automation Provider is a set of Java interfaces and classes which may be accessed locally or remotely to drive POS function. The classes in the automation package implement an object oriented view of the POS application. The package contains classes which model sales transactions, operators, items, tenders, etc.

The method calls within the automation package are designed to be synchronous. All operations are either successful (and return a result if appropriate), or fail by throwing an exception.

The automation provider package is anchored by the POSAutomationProvider interface. Each AEFSession contains an instance of POSAutomationProvider. Operations on the POSAutomationProvider may return instances of other interfaces from the automation package. For example, calling POSAutomationProvider.startTransaction() returns an instance of SalesTransaction. The SalesTransaction interface includes addItem methods which return instances of the Item interface.

The following sequence diagram illustrates retrieving the POSAutomationProvider instance from the AEFSession, and using it to execute a sales transaction.

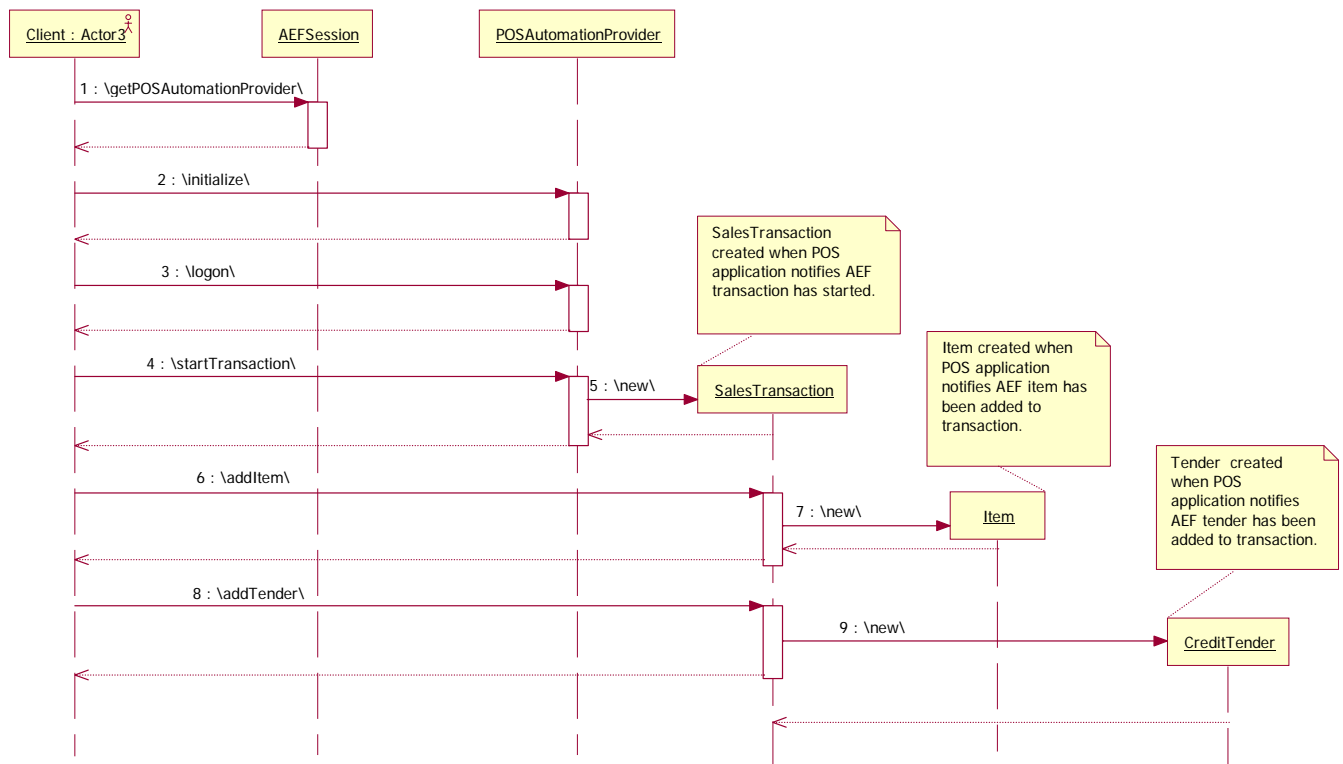


Figure 9 Execute a Transaction Sequence

Error Handling Modes

The automation provider drives input into the POS application, and monitors the results. When the underlying POS application experiences an error which requires operator guidance and/or operator intervention, the result is dependant upon the automation provider's error handling mode.

Automatic Mode

In “automatic” error handling mode, the AEF will make every attempt to complete the requested operation, regardless of the guidance displayed by the POS application. This includes:

- Automatically clearing informational guidance messages.
- If configured, automatically performing operator and manager overrides when required. If the POS application requires a manager number, then a valid manager override number will have to be configured.
- If configured, automatically providing a legal birthdate for age restricted items.

Even with these interventions, there are still many application errors which cannot be circumvented. In these cases, the requested operation cannot complete, and an `AEFException` is thrown back to the caller.

Automatic mode is designed for client applications where there is no “operator” such as a store clerk or cashier. This level of error handling is appropriate for end users who cannot be expected to deal with cashier level guidance. Examples would be web based home shopping, or self checkout.

Default Mode

In “default” error handling mode, the AEF will wait for operator intervention whenever the POS application displays operator guidance messages. The caller is blocked during the time the AEF is waiting for the operator to resolve the guidance.

Note that if default error handling mode is used, the RMI read timeout values specified in the config bundle should be set to zero so that there is no socket read timeout while waiting on the operator. See the *config* bundle property [default.read.timeout](#) on page 173. In addition, all the specific read timeouts should be set to zero.

Setting the Error Handling Mode

The error handling mode may be set statically or dynamically.

To set the error handling mode statically, set the *automation* bundle property [com\ibm\retail\AEF\automation\error.handling.mode](#) on page 159.

To set the error handling mode dynamically, use the [POSAutomationProvider.setProperty](#) method. The following code sets the automation provider error handling mode to “default”.

```
automationProvider.setProperty(POSAutomationProvider.ERROR_HANDLING_MODE, POSAutomationProvider.HANDLE_DEFAULT);
```

Identifier Arguments

Many of the interfaces within the automation package have pairs of methods which differ in the complexity of their arguments. For example, within the `POSAutomationProvider` interface, there is a [logon](#) method which takes an explicit operator id, and an explicit password:

```
public Operator logon(String operatorID, String password) throws RemoteException, AEFException;
```

The code to logon using this method would be:

```
// Logon with operator id = 542, password = 335.  
Operator operator = automation.logon("542","335");
```

There is an equivalent [logon](#) method which takes an instance of [OperatorIdentifier](#).

```
public Operator logon(OperatorID operatorID) throws  
RemoteException, AEFException;
```

The `OperatorIdentifier` interface contains methods to set the operator ID and password.

Each `(obj_type)Identifier` interface is implemented by a class called `(obj_type)IdentifierImpl`. Following the example above, an instance of [OperatorIdentifierImpl](#) could be used as an argument to the `logon` method. Each of the `IdentifierImpl` classes includes constructor signatures which include the commonly required arguments.

The following shows the use of an `OperatorIdentifier` in the `logon` method.

```
// Create an operator id with id = 542, password = 335.  
OperatorIdentifier opID = new OperatorIdentifier("542", "335");  
  
Operator operator = automation.logon(opID);
```

There are two more important points about Identifiers as arguments:

1. Identifier objects as parameters are useful because they derive from the `HashMap` class and therefore allow any key/value pair. This provides for API extension when additional arguments must be included because of application customizations.
2. There is an inheritance structure of Identifier interfaces. If an automation method is defined to take an instance of an Identifier interface, it is valid to pass an instance of an object implementing a subclass of that Identifier interface. In some cases, it is actually necessary to pass a subclass. For example, the [SalesTransaction.addTender](#) method takes a [TenderIdentifier](#) instances. However, a subclass of this interface (such as [CreditIdentifier](#) or [MSRCreditIdentifier](#)) should be passed to determine the type of tender.

The following figure illustrates the hierarchy of Identifier interfaces.

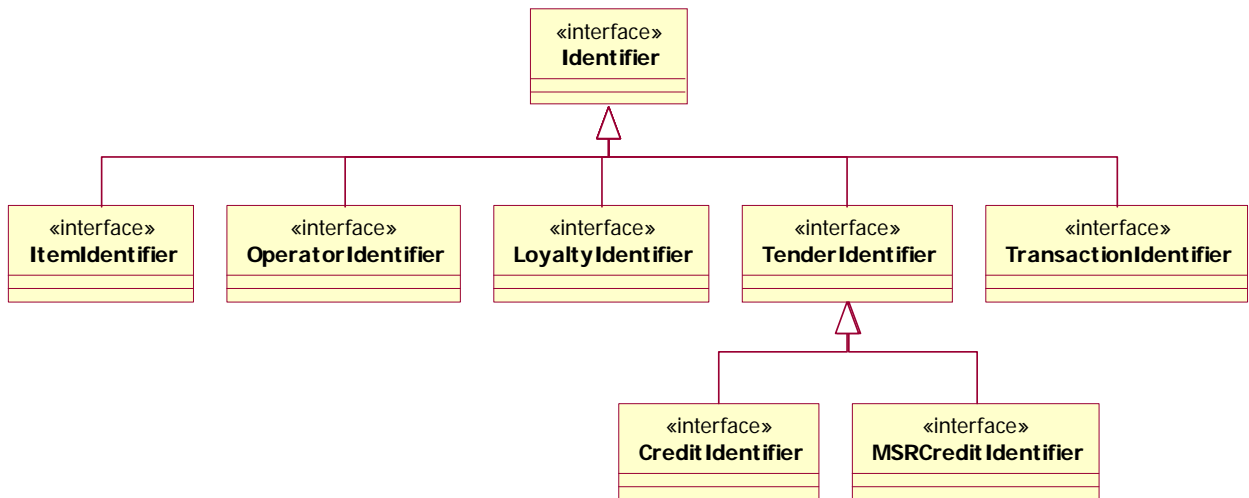


Figure 10 Identifier Interface Hierarchy

Info Objects

The interfaces within the automation package provide for remote access. When an object obtains a reference to a remote object, all the method calls to that object require a roundtrip on the network. To optimize performance, a remote client should minimize the number of remote method calls. To facilitate this, the AEF is architected so that much of the remote object's instance data is encapsulated in a Serializable "info" object.

For example, when an `Operator` instance is returned from a `POSAutomationProvider.logon` call, the [Operator.getInfo](#) method may be called to return an instance of [OperatorInfo](#). The `OperatorInfo` object contains method to retrieve the operator name, ID, and password. The info object allows all three attributes to be returned to the client in a single method call, rather than a separate remote call for each attribute.

The following figure illustrates the Info interfaces and the objects which own their instances.

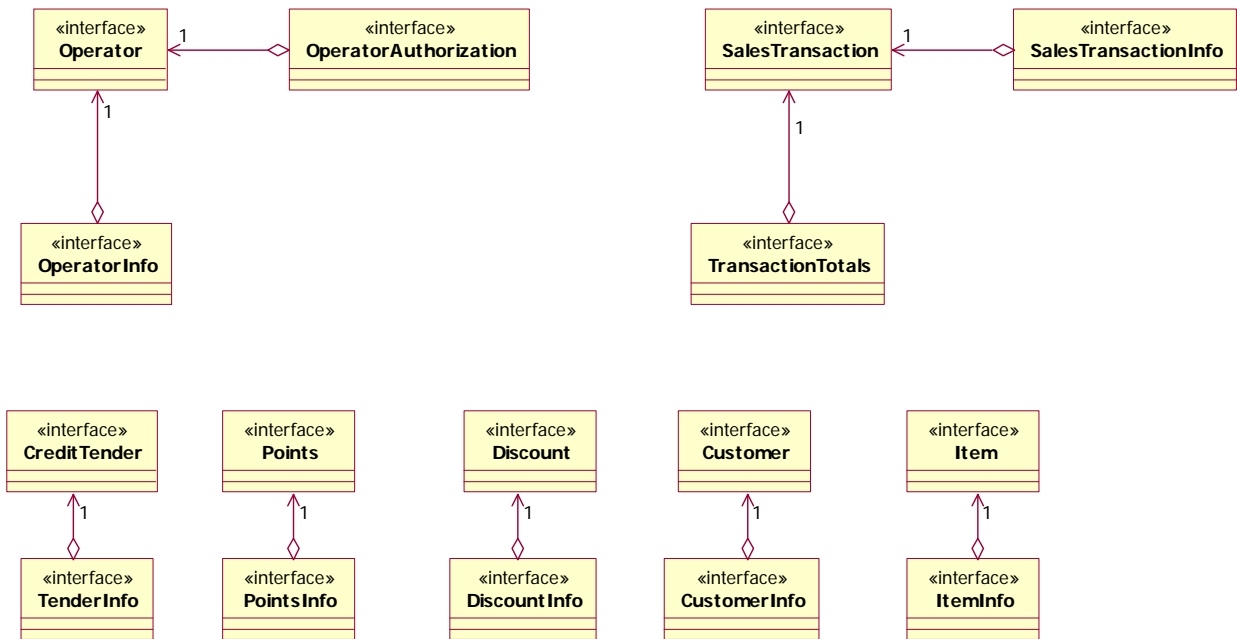


Figure 11 Info Object Interfaces

Configuration Properties

The AEF Automation Provider may be configured via properties in the *automation* bundle.

Property Name	Default Value	Valid Values	Notes
com\\ibm\\retail\\AEF\\automation\\automatic.manager.override	true	true, false	Controls whether the Automation Provider will automatically perform a manager override when required.
com\\ibm\\retail\\AEF\\automation\\automatic.manager.override.number	1	any numeric	For SA & ACE, this is the manager override used for automatic overrides. It must be a valid override number configured in the application.
com\\ibm\\retail\\AEF\\automation\\automatic.operator.override	true	true, false	For SA & ACE, controls whether the Automation Provider will automatically perform an operator override when required.
com\\ibm\\retail\\AEF\\automation\\item.age.restriction	true	true, false	For SA & ACE, controls whether Automation Provider will automatically handle a prompt for

			birthdate for age restricted items. If true, Automation Provider will provide a fixed legal birthdate for the prompt.
com\\ibm\\retail\\AEF\\automation\\automatic.special.signon	true	true, false	For SA & ACE, controls whether Automation Provider will automatically perform a special signon for sessions that are in secure signoff mode.
com\\ibm\\retail\\AEF\\automation\\error.handling.mode	automatic	automatic, default	Determines whether Automation Provider error handling logic will attempt to automatically handle operator guidance (automatic mode), or wait for an operator (default).
com\\ibm\\retail\\AEF\\automation\\error.max.errors.to.handle	10	numeric	Specifies the number of sequential errors the Automation error handler should attempt to deal with before failing the request. This is used in situations where the application may be looping through a set of two or more errors.
com\\ibm\\retail\\AEF\\automation\\automatic.till.exchange	true	true, false	For SA & ACE, determines whether the Automation Provider will perform a till exchange automatically when required by the application.

Table 3 Automation Provider Configuration Properties

POSAutomationProvider

The [POSAutomationProvider](#) interface provides such functions as operator signon and signoff, starting and voiding transactions, etc. Each `AEFSession` contains an instance of a `POSAutomationProvider` object.

Operator

An instance of the [Operator](#) interface is returned from a successful call to the `POSAutomationProvider.logon` method, or by calling [POSAutomationProvider.getOperator](#) if an operator is already logged onto the application.

Each `Operator` instance contains an instance of `OperatorInfo`, and `OperatorAuthorization`. `OperatorInfo` contains the operator ID, name, and password. `OperatorAuthorization` contains application specific values indicating which functions the operator is authorized to perform.

Customer

An instance of the [Customer](#) interface is returned from a successful call to the `SalesTransaction.addCustomerLoyaltyID` method, or by calling `SalesTransaction.getCustomer` if a loyalty ID has already been added to the transaction.

Each instance of `Customer` contains an instance of [CustomerInfo](#). The `CustomerInfo` interface includes information such as customer id, name, points totals and balances, and targeted coupons and messages.

Transaction

An instance (or subclass of) the [Transaction](#) interface is returned from a successful call to the `POSAutomationProvider.startTransaction` method, or by calling [POSAutomationProvider.getTransaction](#) if a transaction is already in progress. For this release, the only type of transactions which are supported are sales transactions. Therefore, calls to `POSAutomationProvider.startTransaction` return the `SalesTransaction` subinterface of the `Transaction` interface.

Each `Transaction` instance contains an instance of `TransactionInfo`. `TransactionInfo` contains the transaction ID (transaction number), transaction date, and a void indicator.

SalesTransaction

The [SalesTransaction](#) interface represents the only type of transaction supported in this release of SI. The `SalesTransaction` interface supports adding, removing, and returning items, adding and removing tenders, adding customer loyalty numbers, and suspending the transaction.

Each `SalesTransaction` instance contains an instance of [SalesTransactionInfo](#) and [TransactionTotals](#). `SalesTransactionInfo` includes a tax exempt indicator. `TransactionTotals` includes the transaction amount, balance due, etc. `TransactionTotals` objects are current only at the time they are retrieved. They include the following transaction amounts: total, subtotal, tax, balance due, foodstamp balance due, foodstamp total, change due, foodstamp change due, coupon total, total items, total coupons, and loyalty savings.

LineItem

The [LineItem](#) interface represents a single item within the transaction. Line items include item sold, coupons, tenders, discounts, and loyalty points. There are subinterfaces of the `LineItem` interface for each of the line item types.

Some of the `SalesTransaction` methods such as `addItem` returns an `ArrayList` of `LineItems`. The client can iterate through the array and use the Java “instanceof” operator to determine each line item’s type. For example, because of linked items and

electronic coupons, a single `addItem` call may result in an `ArrayList` containing multiple `Item` elements, as well as `Coupon` elements. Note that both `Item` and `Coupon` are extensions of the `LineItem` interface type.

The `LineItem` interface includes support for voiding the line item, fetching the `LineItemInfo` object, and fetching the item's identifier.

The [`LineItemInfo`](#) includes a description, receipt lines, a void indicator, refund indicator, and deposit indicator. Usually, the instance of `LineItem` will be an instance of a subinterface of `LineItem` (such as `Coupon`, `Item`, etc.). Therefore, the `LineItemInfo` will actually be an instance of a subinterface such as `CouponInfo` or `ItemInfo`.

Item

The [`Item`](#) interface represents a purchase item within the transaction. The `Item.getInfo` method returns an instance of [`ItemInfo`](#). Each `Item` contains an instance of `ItemInfo`.

The `ItemInfo` interface includes item identifier, item pricing information, quantity, weight, age restriction, foodstamp indicator, WIC indicator, restricted sales periods, item repeat indicator, and tax information.

Coupon

The [`Coupon`](#) interface represents a store or manufacturer coupon within the transaction. The `Coupon.getInfo` method returns an instance of [`CouponInfo`](#). Each `Coupon` contains an instance of `CouponInfo`.

The `CouponInfo` interface includes a coupon type, as well as sharing many of the `Item` attributes.

Discount

The [`Discount`](#) interface represents a line item or transaction discount within the transaction. The `Discount.getInfo` method returns an instance of [`DiscountInfo`](#). Each `Discount` contains an instance of `DiscountInfo`.

The `DiscountInfo` interface includes information on the type of discount, the discount rate or amount, the discount reason, a taxability indicator, and whether the discount applies to the entire transaction or a specific item.

Points

The [`Points`](#) interface represents the awarding or redemption of customer loyalty points within a transaction. `Points` objects are `LineItems` and may be automatically generated by the POS application in response to selling an item. Each `Points` object contains an instance of [`PointsInfo`](#).

The `PointsInfo` interface includes information such as the type of points, whether the points are awarded or redeemed, and the number of points involved.

Tender

The [Tender](#) interface represents a tender `LineItem` within the transaction. For this release, the only tenders supported by the `SalesTransaction.addTender` method are credit tenders. An instance of `CreditIdentifier` or `MSRCreditIdentifier` should be passed to the `SalesTransaction.addTender` method. If the client has access to MSR track data, then the `MSRCreditIdentifier` should be used. Each `Tender` object contains a [TenderInfo](#) instance.

The `TenderInfo` interface includes the tender amount.

CreditTender

For this release of SI, the only tender supported for the `SalesTransaction.addTender` method are credit tenders. This means that the object returned from a successful `addTender` call will be instances of the [CreditTender](#) interface. Each `CreditTender` object contains an instance of [CreditInfo](#).

The `CreditInfo` interface includes the tender amount and account number.

Using Remote References

The Automation Provider objects listed above are remote objects. If a remote client holds a reference to an object instance, the object cannot be garbage collected in the host JVM. If a client application is illbehaved or exhibits a memory leak, the host JVM can experience a memory leak because the remote objects referenced by the client cannot be garbage collected until the client dereferences the objects.

An example of this can be illustrated by a remote client program which attaches to a real terminal session. The remote client gets a reference to the `SalesTransaction`. The operator on the real register then ends the transaction. There are no more references to the `SalesTransactions` on the register, but the `SalesTransaction` cannot be garbage collected until the client dereferences the remote reference by nulling the `SalesTransaction` reference, letting the reference go out of scope, terminating the application, or losing network connectivity to the host JVM.

Automation API “Under the Covers”

This section describes how the Automation Provider API is implemented “under the covers”. An understanding of the workings of the Automation Provider is required in order to customize its behavior. Modifications or extensions to the AEF may be necessary to “AEF enable” user modifications to the POS application.

Action Classes

The subset of Automation Provider methods which drive input into the POS applications are implemented via “action” classes. The Automation Provider methods which make use of action classes will include the following line in the class javadoc:

Action ID = *ActionKey*

The *ActionKey* can be looked up in the “class” bundle (see page 160) to determine the class used to implement the action. The following table also lists the AEF “action” classes.

Base Action Classes
AEFActionImpl.java
InitializeActionImpl.java
SwipeMSRActionImpl.java
ACE Action Classes
ACEActionImpl.java
ACEApplyDelayedCouponsActionImpl.java
ACECancelPreviousEntryActionImpl.java
ACECustomerEntryActionImpl.java
ACEEPCreditTenderActionImpl.java
ACEForcedLogoffActionImpl.java
ACEItemEntryActionImpl.java
ACEItemReturnActionImpl.java
ACEItemVoidActionImpl.java
ACELogoffActionImpl.java
ACELogonActionImpl.java
ACEManagerOverrideActionImpl.java
ACEOperatorOverrideActionImpl.java
ACERetrieveSuspendedTransactionListActionImpl.java
ACERetrieveTransactionActionImpl.java
ACESetTrainingModeOnActionImpl.java
ACEStartTransactionActionImpl.java
ACESuspendTransactionActionImpl.java
ACEVoidTransactionActionImpl.java
GSA Action Classes
GSAActionImpl.java
GSACancelPreviousEntryActionImpl.java
GSAClearActionImpl.java
GSAEFTCreditTenderActionImpl.java
GSAForcedLogoffActionImpl.java
GSAItemEntryActionImpl.java
GSAItemReturnActionImpl.java
GSAItemVoidActionImpl.java
GSALogoffActionImpl.java
GSALogonActionImpl.java
GSAManagerOverrideActionImpl.java
GSARetrieveSuspendedTransactionListActionImpl.java
GSARetrieveTransactionActionImpl.java
GSASetTrainingModeOnActionImpl.java
GSAStartTransactionActionImpl.java
GSA suspendTransactionActionImpl.java
GSAVoidTransactionActionImpl.java
SA Action Classes
SAActionImpl.java
SAApplyDelayedCouponsActionImpl.java
SACancelOverrideActionImpl.java
SACancelPreviousEntryActionImpl.java
SAClearActionImpl.java
SACustomerEntryActionImpl.java

SAEFTCreditTenderActionImpl.java
SAEnterStandAloneActionImpl.java
SAForcedLogoffActionImpl.java
SAltemEntryActionImpl.java
SAltemReturnActionImpl.java
SAltemVoidActionImpl.java
SALogoffActionImpl.java
SALogonActionImpl.java
SAManagerOverrideActionImpl.java
SAOperatorOverrideActionImpl.java
SARetrieveSuspendedTransactionListActionImpl.java
SARetrieveTransactionActionImpl.java
SASetTrainingModeOnActionImpl.java
SAStartTransactionActionImpl.java
SASuspendTransactionActionImpl.java

Figure 12 Action Classes

When an action is constructed, it is passed an instance of [ActionRequest](#) which contains a HashMap of all the arguments necessary to perform the action. The action performs its function when its `performAction` method is called. In general, the steps taken in the `performAction` method are:

- Check to see if the POS application is currently in an error state which must be handled prior to performing the action.
- Check any preconditions that must hold prior to performing the action. Examples are actions which require the POS application be in a specific state or substate in order to perform the action.
- Create a [SimpleKeySequenceActionImpl](#) based on arguments from the `ActionRequest`. The `SimpleKeySequenceActionImpl` will send a key sequence to the POS application.
- Define the expected results of the key sequence. This includes “good” and “bad” conditions. Good conditions result when the POS application reacts to the key sequence as expected. Bad conditions result when the POS application reacts to the key sequence by generating an error requiring operator guidance, or other results which require AEF error handling.
- Use a [ConditionLock](#) to perform the `SimpleKeySequenceActionImpl` and to wait for one of the good or bad conditions to be observed. If a “bad” condition is detected, pass control to the AEF error handler to resolve.
- If the action is supposed to return an object or objects (such as an `Operator` or `LineItems`), then it uses an [ObjectDetectorLock](#) to wait until the objects are detected.

See “Appendix E. `SALogonActionImpl` Source Example” on page 358 for a detailed discussion of the `SALogonActionImpl` class.

Property Conditions

A property condition is a logical condition predicated on a `POSDDataProvider` property. Property conditions evaluate to true or false depending on the value of a `POSDDataProvider` property. There are property condition classes which check for value equality, value greater than, value less than, value not equal to, etc.

The action classes discussed above use property conditions to define “good” and “bad” results when injecting input into the POS application. For example, we may want to define a “good” condition that the POS application is in state 10 and is either in substate 1001 or 1008. Since property conditions are always defined as arrays in Java, we could represent that good condition with the declaration:

```
Condition substateConditions[] =
{
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_SUB_STATE,
                                "1001"),
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_SUB_STATE,
                                "1008")
};

Condition andConditions[] =
{
    new OrCondition(substatesConditions),
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_STATE,
                                "10")
};

Condition goodConditions[] =
{
    new AndCondition(andConditions)
};
```

The condition array “goodConditions” can be passed to a ConditionLock to define an expected “good” result of injecting POS input.

All conditions within the condition array are assumed to be logically “or’ed” together. In the example above, the final array “goodConditions” has only a single element. The following example declares a condition array which will be true if any of the elements are true because the array elements are logically “or’ed”.

```
Condition badConditions[] =
{
    // All 4 conditions are “or’ed” so this set of conditions
    // is true if any of the substates (10333, 10336, 10339,
    // 10343) are observed.
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_SUB_STATE,
                                "10333"),
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_SUB_STATE,
                                "10336"),
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_SUB_STATE,
                                "10339"),
    new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
                                POSDeviceProperties.POS_SUB_STATE,
                                "10343"),
}
```

The AEF includes Java classes which support the following property value comparisons:

Java Class Name com.ibm.retail.AEF.automation package	Operation	Example
PropertyEqualsCondition	Checks a property value for equality with the specified value. Specified value may be a string or a number.	<p>A property condition which determines when the POSDataProvider property (property category =POS_DEVICE, property name=subState) equals "1008" could be defined as :</p> <pre>PropertyEqualsCondition("POS_DEVICE", "subState", "1008")</pre> <p>Another way to specify this same condition is :</p> <pre>PropertyEqualsCondition(POSDeviceProperties.CATEGORY, POSDeviceProperties.POS_SUB_STATE, Substate.getSubstate("SELECT_PROCEDURE"))</pre>
PropertyNotEqualsCondition	Checks a property for non-equality with the specified value. Specified value may be a string or a number.	<p>A property condition which determines when the current state is not the clear state (state 1) could be defined as:</p> <pre>PropertyNotEqualsCondition(POSDeviceProperties.CATEGORY, POSDeviceProperties.POS_STATE, State.getState("CLEAR"))</pre>
PropertyGreaterThanCondition	Checks a property value to see if it is greater than the specified value. The property value must be able to be converted to a numeric value.	<p>A property condition which indicates that the number of customer loyalty points is greater than 5000 could be defined as:</p> <pre>PropertyGreaterThanCondition(PointsProperties.CATEGORY, PointsProperties.TOTAL, 5000)</pre>
PropertyGreaterOrEqualCondition	Checks a property value to see if it is greater than or equal to the specified value. The property value must be able to be converted to a numeric value.	<p>A property condition which indicates when the transaction total is greater than or equal to zero could be defined as:</p> <pre>PropertyGreaterOrEqualCondition(TransactionTotalsProperties.CATEGORY, TransactionTotalProperties.TOTAL, 0)</pre>
PropertyLessThanCondition	Checks a property value to see if it is less than the specified value. The property value must be able to be converted to a numeric value.	<p>A property condition which indicates when the scale weight drops below 1 could be defined as:</p> <pre>PropertyLessThanCondition(POSDeviceProperties.CATEGORY, POSDeviceProperties.SCALE_WEIGHT VALUE, 1)</pre>
PropertyLessOrEqualCondition	Checks a property value to see if it is less than or equal to the specified value. The property value must be able to be converted to a numeric value.	<p>A property condition which indicates when the transaction total is less than or equal to 5.25 could be defined as:</p> <pre>PropertyLessOrEqualCondition(TransactionTotalsProperties.CATEGORY, TransactionTotalProperties.TOTAL, 5.25)</pre>
PropertyContainsCon	Check a property value to	A property condition which indicates when the

dition	see if it contains the specified substring.	first line of the line display contains the string "ERROR" could be defined as: PropertyContainsCondition (POSDeviceProperties.CATEGORY, POSDeviceProperties.ANPROMPT_LINE1, "ERROR")
PropertyNotContainsCondition	Indicates when a property value does not contain the specified substring.	A property condition which indicates when the first line of the customer display does not contain the string "WELCOME" could be defined as: PropertyContainsCondition (POSDeviceProperties.CATEGORY, POSDeviceProperties.CUST_PROMPT_LINE1, "WELCOME")
PropertyContainsAtIndexCondition	Indicates when a property contains the specified substring at the specified (0 based) index.	A property condition which indicates when the first line of the customer display contains the substring "CREDIT" starting at the third character of the line could be defined as: PropertyContainsAtIndexCondition (POSDeviceProperties.CATEGORY, POSDeviceProperties.CUST_PROMPT_LINE1, "CREDIT" , 2)
PropertyNotContainsAtIndexCondition	Indicates when a property does not contain the specified substring at the specified (0 based) index.	A property condition which indicates when a customer's loyalty number expiration date does not contain "04" in the 4 th position could be defined as : PropertyNotContainsAtIndexCondition (CustomerProperties.CATEGORY, CustomerProperties.ID_EXPIRATION_DATE, "04" , 3)
PropertyRegexMatchCondition	Indicates when a property value matches the specified regular expression.	A property condition which indicates when the first line of the customer display starts with the letter "B" followed by three digits could be defined as: PropertyRegexMatchCondition (POSDeviceProperties.CATEGORY, POSDeviceProperties.CUST_PROMPT_LINE1, "B\\d\\d\\d")
PropertyRegexNotMatchCondition	Indicates when a property value does not match the specified regular expression.	A property condition which indicates when the last item scanned does not begin with a letter could be defined as: PropertyRegexNotMatchCondition (POSDeviceProperties.CATEGORY, POSDeviceProperties.LAST_SACN_LABEL, "[a-zA-z]")
AndCondition	Evaluates true when all the child conditions evaluate true.	If badConditions is an array holding property conditions, a new AndCondition containing the badConditions array as children could be declared as:

		<code>AndCondition(badConditions)</code>
<u>OrCondition</u>	Evaluates true when any of the child conditions evaluate true.	<p>If goodConditions is an array holding property conditions, a new OrCondition containing the goodConditions array as children could be declared as:</p> <p><code>OrCondition(goodConditions)</code></p> <p>Note: Passing an array of Condition objects to a ConditionLock causes the array element conditions to be logically “or’ed”, so it may not be necessary to create an explicit “OrCondition”.</p>
<u>AndThenCondition</u>	Similar to the AndCondition, except that initially, only the first child condition is eligible for evaluation. If the first child condition evaluates to true, then the second child condition becomes eligible for evaluation, and so on. When all children conditions evaluate to true, the overall AndThenCondition evaluates to true.	<p>If badConditions is an array holding property conditions, a new AndThenCondition containing the badConditions array as children could be declared as:</p> <p><code>AndThenCondition(badConditions)</code></p>

Table 4 Property Condition Classes

Bad Conditions

Since a common set of bad conditions is often used repeatedly by the actions, they have been made available via the following method calls:

```
// Standard bad conditions
BadConditionsImpl.getInstance().getBadConditions();
// Bad conditions for EFT/EPS
BadConditionsImpl.getInstance().getExtendedBadConditions();
```

The set of bad conditions returned from each method is dependent on the POS application, and can be viewed in the following source modules:

[ACEBadConditionsImpl.java](#)
[GSABadConditionsImpl.java](#)
[SABadConditionsImpl.java](#)

ConditionLock

The [ConditionLock](#) is used to inject keyboard or MSR input into the POS application, and to wait for one of a set of specified property conditions to be observed. The conditions are usually defined as a set of “good” conditions, and a set of “bad” conditions.

The following code fragment shows a call to a ConditionLock to cause the key sequence to be injected. The return code will indicate which of the good or bad conditions was observed.

```
HashMap args = new HashMap();
```

```

args.put("%0", password);
args.put("SEQUENCE_ID", "password");

ConditionLock lock = new ConditionLock();

AEFAction keySequenceAction = (AEFAction)
    (actionFactory.makeAction(new
        ActionRequest("SimpleKeySequenceAction",
            args)));

retVal= lock.performActionAndWait("wait-for-item-entry-state",
    keySequenceAction,
    goodConditions,
    BadConditionsImpl.
        getInstance().
        getBadConditions(),
    getTimeout());

```

In this example, a `SimpleKeySequenceAction` is created which will send the key sequence to enter an operator password. The `ConditionLock` is used to perform the action. The first argument of the `performActionAndWait` method is a string used for trace which is logged if the call times out. The key sequence is passed, along with the good and bad conditions. Finally a timeout value is passed. Here, we use the default timeout which is configured in the “config” bundle (see page 171).

The `performActionAndWait` call will either return with an exception, or an integer value. The call will return with an exception if there is a timeout waiting for any of the conditions to be observed, or if there is some other error. If one of the conditions is observed, an integer will be returned. This integer will indicate which condition evaluated to true. An index of zero or greater indicates one of the “good” conditions. The first element in the “good” condition array is “0”, the second is index “1”, and so on. A negative return value indicates one of the “bad” conditions was observed. The first element in the “bad” condition array is index “-1”, the second is index “-2”, and so on. If a bad condition is observed, the action class usually passes control to the `AEFErrorHandler` to attempt to resolve the error.

ObjectDetectorLock

The [ObjectDetectorLock](#) is used in conjunction with an object detector to block the calling thread until a new instance of a specific object type is detected. The `waitForNewObject` method will block the calling thread until a new object is detected, or until a timeout occurs. The `waitForNewObjectOrError` method adds the capability to unblock if one of the specified error conditions is observed. The automation action classes such as the `SALogonActionImpl` class use instances of the `ObjectDetectorLocks` to wait for the item to be returned, such as a new `Operator` instance.

The object detectors themselves may be retrieved from the `AEFSession` instance.

AEF Error Handling

When an error condition (as defined by a set of “bad” conditions) is detected as a return value from a `ConditionLock` within an action class, the action usually creates an

instance of the [AEFErrorHandler](#) class. When created, the AEFErrorHandler attempts to determine which error helper should be created to process the error.

The error helper's job is to either resolve the error, or throw an exception if the error cannot be resolved.

The error helper class to used is defined in the "error" bundle as described on page 177. For a particular error key, there is a set of entries in the "error" bundle. One of the entries will be the error helper class key. An error code and extended error code is also defined for each error key. These are the error codes that will be used if the error helper cannot resolve the error and must throw an exception.

The following lines from the SA version of apperror.properties correspond to the "B069" error key.

```
# B069 Tenders/Coupons Must be Returned

B069_ADDITIONAL_TEXT=A transaction has been voided after coupons
or other tenders have been accepted.
B069_HANDLING_CLASS_KEY=InformationalErrorHandler
B069_ERROR_CODE=NONE
B069_EXTENDED_ERROR_CODE=NONE
```

The error helper class key which corresponds to an error key of "B069" is "InformationalErrorHandler". The error helper key "InformationalErrorHandler" is looked up in the "classes" bundle to determine the error helper class. In this case, the error helper class is `com.ibm.retail.AEF.util.SAErrorHandlerInformational`.

The informational error helper will clear the error if the AEF error handling mode is "automatic". Alternatively, the informational error helper will wait for the operator to clear the error if the AEF error handling mode is "default".

Determining the Error Key

When an AEFErrorHandler is created, it must determine the error key which is used to look up the error helper class key as described above. Depending on the POS application, the error key is usually based on the application state or substate, or an **Annn** or **Bnnn** guidance number on the operator display. The algorithms for determining the error key are described below.

ACE Error Key Algorithm

When an AEFErrorHandler is created for ACE, the current ACE substate is used to lookup the error key in "appsubstates.properties" (see page 269). For example, if the current ACE substate is 10069, then the error key is "B069" as specified by the following line in "appsubstates.properties":

```
# B069 TENDERS/COUPONS MUST BE RETURNED
10069=B069
```

If the error key does not exist in "appsubstates.properties" or in "apperror.properties", then the error key is set to "APP_DEFAULT".

GSA Error Key Algorithm

When an AEFErrorHelper is created for GSA, if the operator display contains an **A***nnn* guidance number, then the guidance number is used as the error key. If this error key isn't listed in "apperror.properties", then the entire first line of the operator display is used to lookup a descriptor number. For example, if the first line of the operator display matches the descriptor number 35 as defined in GSA, then the error key is "BASE_DESC35". If this error key is not listed in "apperror.properties", then the current application state is appended onto "STATE_" to form the error key. For example, if the current GSA state is "20", then the error key is "STATE_20". If this error key is not listed in "apperror.properties", then the error key is set to "APP_DEFAULT".

SA Error Key Algorithm

When an AEFErrorHelper is created for SA, if the operator display contains a **B***nnn* guidance number, then the guidance number is used as the error key. If this error key isn't listed in "apperror.properties", then the entire first line of the operator display is used to lookup a descriptor number. For example, if the first line of the operator display matches the EFT descriptor number 20 as defined in SA, then the error key is "EFT_DESC20". If this error key is not listed in "apperror.properties", then the current application substate is used as the error key. If this error key is not listed in "apperror.properties", then the error key is set to "APP_DEFAULT".

Error Helper Classes

The following table lists the error helper classes available in the AEF. Instructions for writing a new error helper are included in Chapter 3.

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
ACEErrorHelperB004 SAErrorHelperB004	Used to handle the "B004 Managers Key is Required" prompt. If the error handling mode is set to "automatic", and the Automation Provider is configured for automatic manager override, the error helper clears the error, and sets the keylock position to manager mode. If the error handling mode is "default", the error helper waits for the operator to clear the error, then an exception is thrown.	ACE, SA
ACEErrorHelperB005 SAErrorHelperB005	Used to handle the "B004 Remove Managers Key" prompt. If the error handling mode is set to "automatic", and the Automation Provider is configured for automatic manager override, the error helper clears the error, and restores the keylock position if necessary. If the error handling mode is "default", the error helper waits for the operator to clear the error, then an exception is thrown.	ACE, SA
ACEErrorHelperB013	Used to handle the "B013 CLOSE	ACE, SA

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
SAErrorHelperB013	THE CASH DRAWER” prompt. In both “automatic” and “default” error handling modes, the error helper waits for the cash drawer to be closed. If the cash drawer is emulated, it will close automatically. If the cash drawer is not emulated, the operator must close it.	
ACEErrorHelperB040 SAErrorHelperB040	Used to handle the “B040 FILE ACCESS FAILED” message. If the error handling mode is set to “automatic”, the error helper clears the error, performs a manager override if the error is eligible to be overridden, and the Automation Provider is configured for automatic override. Finally, if the application prompts to enter standalone, the error helper will perform the key sequence to enter standalone mode. If the error handling mode is “default”, the error helper waits for the operator to clear the error. The error helper then waits for the operator to perform or cancel a manager override if the override prompt appears. Finally, the error helper waits for the operator to perform the key sequence to enter standalone if prompted to do so. In either error handling mode, an exception is thrown.	ACE, SA
ACEErrorHelperB085 SAErrorHelperB085	Used to handle the “B085 TILL EXCHANGE REQUIRED” message. If the error handling mode is set to “automatic” and the Automation Provider is configured to perform automatic till exchanges, then the error helper clears the error, and sends the till exchange key sequence. If the error handling mode is “default”, the error helper clears the error, and throws an exception.	ACE, SA
SAErrorHelperB113	Used to handle the “B113 REMOVED FRANKED TENDER” message. For both “automatic” and “default” error handling modes, the error helper waits for the document to be ejected from the printer.	SA
SAErrorHelperB537	Used to handle the “B537 NO	SA

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
	STAND-IN FOR THIS SERVICER” message. If the error handling mode is “automatic”, the error helper clears the error and throws an exception. If the error handling mode is “default”, the error helper waits for the operator to clear the error, then throws an exception.	
SAErrorHelperB608	Used to handle the “B608 ASK FOR CARD CLEAR TO CONTINUE” prompt. There are two reasons for getting this message. One is a reminder to get the customer's card when the total key is pressed. The other is when a coupon is entered that is limited to preferred customers. If the error handling mode is “automatic”, the error helper clears the error. If the prompt was generated in response to a coupon that is limited to preferred customers, the error helper throws an exception. If the error handling mode is “default”, the error helper waits for the operator to clear the error. If the prompt was generated in response to a coupon that is limited to preferred customers, the error helper throws an exception.	SA
ACEErrorHelperB676 SAErrorHelperB676	Used to handle the “B676 ACTIVE SUSPENDS, CLEAR OR CONTINUE” prompt. If the error handling mode is “automatic”, the error helper clears the error to allow the signoff. If the error handling mode is “default”, the error helper waits for the operator to clear the error to allow the signoff.	ACE, SA
AcceptOrVoidTender	Used to handle the “ENTER TO ACCEPT OR CLEAR TO VOID” tender prompt in GSA. If the error handler is in “automatic” mode, the error helper sends the <ENTER> key to accept the tender. If the error handler is in “default” mode, the error helper waits for the operator to handle the prompt. If the operator clears the tender, an exception is thrown.	GSA

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
AccountNumberErrorHandler	<p>GSA: Used to handle GSA state 51. In "automatic" mode, the error helper retrieves the account number from the tender identifier and sends it. If none is found, an exception is thrown. In "manual" mode, the AEF waits for the operator to enter an account number.</p> <p>SA: Used to handle "B048 ACCOUNT NUMBER NEEDED" guidance message. In "automatic" mode, this is a fatal error, the guidance is cleared, and an exception is thrown. In "manual" mode, the AEF waits for the operator to enter an account number.</p>	GSA, SA
AgeRestrictedHelper	<p>ACE: Used to handle date of birth prompt for age restricted items.</p> <p>GSA: Used to handle "A399 AGE RESTRICTED ITEM" guidance message.</p> <p>SA: Used to handle "B107 AGE CHECK NEEDED" guidance.</p> <p>In "automatic" mode, if the Automation Provider is configured to handle age restricted items automatically, a hard coded date of birth will be entered to satisfy the age prompt. If the Automation Provider is not configured to automatically handle age restricted items, or the error handling mode is "default" an exception will be thrown.</p>	ACE, GSA, SA
AppDefaultErrorHandler	<p>This is the error helper which is used whenever an error key cannot be located in "apperror.properties". An exception is always thrown. This usually indicates that an error helper should be configured.</p>	ACE, GSA, SA
AuthorizationCodeErrorHandler	<p>Used to handle the "ENTER AUTHORIZATION CODE" prompt. In either "automatic" or "default" error handling modes, the error helper uses the authorization code from the tender identifier as input</p>	ACE

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
	for a key sequence to satisfy the prompt.	
BaseDescriptor449ErrorHandler	This error helper may be used when the error handler encounters a blank line on the operator display. The error helper waits for the line display to change.	GSA
CardValidationNumberErrorHandler	Used to handle the "ENTER CARD VALIDATION NUMBER" prompt. For "automatic" or "default" error handling mode, the error helper will extract the card validation number from the TenderIdentifier. If none is provided, an exception is thrown.	ACE
DuplicateReceiptHelper	SA: Used to handle the "ENTER FOR DUPLICATE CLEAR TO BYPASS" prompt.	SA
EnterAuthorizationErrorHandler	Used to handle state 6 in GSA. If the error handling mode is "automatic", the error helper sends a hard coded "19" as the authorization code. If the error handling mode is "default", the error helper waits for the operator to enter an authorization number.	GSA
ExpirationDateErrorHandler	Used to handle the "ENTER EXPIRATION DATE" prompt. For "automatic" or "default" error handling mode, the error helper will extract the expiration date from the TenderIdentifier. If none is provided, an exception is thrown.	ACE
FatalClearErrorHandler	Used for errors that are always non-recoverable. The error requires two clear keys to clear the error. In "automatic" error handling mode, the error helper sends the clear keys. In "default" error handling mode, the error handler waits for the operator to clear the error. In either mode, an exception is thrown after the error is cleared.	ACE, SA
FatalErrorHandler	Used for errors that are always non-recoverable. In "automatic" error handling mode, the error is cleared by the error helper. In "default" error handling mode, the error helper waits for the operator to clear the error. In either mode, an exception is thrown after the error is cleared.	ACE, GSA, SA

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
InformationalErrorHandler	Used for guidance messages that are informational in nature. In “automatic” error handling mode, the error helper clears the guidance. In “default” mode, the error helper waits for the operator to clear the guidance.	ACE, GSA, SA
InsertDocumentHelper	Used to handle the “INSERT DOCUMENT FOR FRANKING” prompt. In “automatic” and “default” error handling mode, the error helper will wait for a document to be detected at the printer document station.	GSA
ItemPromptErrorHandler	Used during item entry sequences to handle prompts for additional data such as item price, item weight, and volume. In “automatic” error handling mode, the prompt is cleared which terminates the item entry. An exception is thrown to indicate to the client which argument must be added to the “addItem” method call. In “default” error handling mode, the error helper waits for the operator to satisfy the prompt, or cancel the item. If the operator cancels the item, an exception is thrown.	ACE, SA
LevelOverrideErrorHandler		
OriginalSalespersonRequiredErrorHandler	Used to handle state 20 “ORIGINAL SALESPERSON REQUIRED”. In both “automatic” and “default” error handling modes, the error helper will attempt to send the original salesperson id key sequence.	GSA
OverrideErrorHandler	Used when the application is giving guidance that a manager or operator override is required to complete the operation. In “automatic” error handling mode, the error helper checks to see if the Automation Provider is configured to perform the override automatically. If configured for automatic override, the error helper clears the guidance and performs the override. In “default” error handling mode, the error helper waits for the operator to clear the guidance and perform the override, or cancel the operation. If the	ACE, GSA, SA

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
	operator cancels the operation, an exception is thrown.	
PriceRequiredErrorHandler	Used to handle the "A109 PRICE IS REQUIRED" prompt. In "automatic" or "default" error handling modes, the error helper will send the price sequence if a price was included in the ItemIdentifier. If no price was included in the Identifier and the error helper is in "automatic" mode, an exception is thrown. If no price was included in the Identifier and the error helper is in "default" mode, the error helper waits for the operator to enter a price.	GSA
PrintBypassErrorHandler	Used to handle the "ENTER TO PRINT OR CLEAR TO BYPASS" prompt. If the error handling mode is "automatic", the error helper waits for the document to be inserted, then sends the <ENTER> key to print the document. If the error handling mode is "default", the error helper waits for the operator to press <ENTER> or <CLEAR>.	GSA
PrinterErrorHandler	Used to handle printer errors. The error helper enters a loop, waiting for the printer problem to be corrected.	ACE, GSA, SA
PrinterPaperOutErrorHandler	Used to handle "out of paper" printer errors. The error helper waits for the printer error to be resolved by the operator.	GSA, SA
PromptErrorHandler		
QuantityErrorHandler	Used to handle state 49 where a quantity is required for an item. If the error handling mode is "automatic" or "default", the error helper will use the quantity provided in the ItemIdentifier. If no quantity was provided in the ItemIdentifier, an exception is thrown.	GSA
QuantityOrPriceErrorHandler	Used to handle the a prompt for quantity or price. If the error handling mode is set to "automatic", the error helper will send the price or quantity sequence if the price or quantity was included in the ItemIdentifier. If not, an exception is thrown. If the error handler mode is "default", the error helper waits for the operator to satisfy the	GSA

Error Helper Key	Error Helper Behavior	Implementation Classes Available For Applications
	prompt for price or quantity.	
RemoveDocumentHelper	Used to handle the "REMOVE DOCUMENT" prompt. In both "automatic" and "default" error handling modes, the error helper waits for the document to be removed from the document insert station of the printer.	GSA
RemoveReceiptErrorHandler	Used to handle the "REMOVE RECEIPT FOR CUSTOMER" prompt. In "automatic" error handling mode, the error helper clears the error. In "default" error handling mode, the error helper waits for the operator to clear the error.	SA
SAErrorHandlerSignature	Used to handle the "VERIFY SIGNATURE" prompt. If the error handling mode is "automatic", the error helper sends the key sequence to verify the signature. If the error handling mode is "default", the error helper waits for the operator to enter the key sequence to verify the signature.	SA
TerminalDisabledErrorHandler	Used to handle the "TERMINAL DISABLED" message. In "automatic" error handling mode, the error helper clears the error, then throws an exception. In "default" error handling mode, the error helper waits for the operator to clear the error, then an exception is thrown.	GSA
VoucherNumberErrorHandler	Used to handle the "ENTER VOUCHER NUMBER" prompt. For "automatic" or "default" error handling mode, the error helper will extract the voucher number from the TenderIdentifier. If none is provided, an exception is thrown.	ACE

Table 5 Error Helpers

Data Provider

The purpose of the Data Provider is to publish events which represent changes in the POS application, and to maintain current property values which represent the current state of the POS application. Clients may register to listen for Data Provider events such as `TransactionStatusEvents`, `ItemSalesEvents`, `TransactionTotalsEvents`, `OperatorEvents`, etc. Clients may also query or set individual property values. When a Data Provider property value is changed, an [AEFPropertyChangeEvent](#) is fired to any registered listeners.

Properties

Data Provider properties are key/value pairs. Each property has a category (type string), a property name (type string), and a value (usually type string).

The category can be any sting value, but there are a set of predefined categories which are listed in the following table.

Category	Category Name	Category Name Constant	Contains Properties
CouponProperties	"COUPON"	<code>CouponProperties.CATEGORY</code>	Individual properties from the last <code>CouponEvent</code> .
CustomerProperties	"CUSTOMER"	<code>CustomerProperties.CATEGORY</code>	Individual properties from the last <code>CustomerEvent</code> .
DiscountProperties	"DISCOUNT"	<code>DiscountProperties.CATEGORY</code>	Individual properties from the last <code>ItemSalesEvent</code> .
ItemSalesProperties	"ItemSales"	<code>ItemSalesProperties.CATEGORY</code>	Individual properties from the last <code>ItemSalesEvent</code> .
LineItemProperties			Individual properties from the last line item event (of any type).
OperatorAuthorizationProperties	"opAuth"	<code>OperatorAuthorizationProperties.CATEGORY</code>	Each property in this category indicates a POS function. The property value is "true" or "false" to indicate whether the operator is authorized to execute the function.
OptionsProperties	"OPTIONS"	<code>OptionsProperties.CATEGORY</code>	Properties related to POS application options settings.
PointsProperties	"POINTS"	<code>PointsProperties.CATEGORY</code>	Individual properties from the last <code>PointsEvent</code> .
POSDataProperties			
POSDeviceProperties	"POS_DEVICE"	<code>POSDeviceProperties.CATEGORY</code>	Properties related to POS I/O devices, such as the contents of the line displays, last labels scanned, cash drawer

Category	Category Name	Category Name Constant	Contains Properties
			state, current application state and substate, etc.
StoreOptionsProperties			Properties related to store options personalization of the POS application.
StoreProperties	"STORE"	StoreProperties.CATEGORY	Properties such as store name, division, address, phone numbers, etc.
TenderProperties	"TENDER"	TenderProperties.CATEGORY	Properties related to the last tender taken such as type, variety, description, amount, etc.
TerminalOptionsProperties			Properties related to terminal options personalization of the POS application.
TransactionStatusProperties	"transactionStatus"	TransactionStatusProperties.CATEGORY	Properties such as transaction id, date, time, category, voided indicator, discounts allowed indicator, void allowed indicator.
TransactionTotalsProperties	"transactionTotals"	TransactionTotalsProperties.CATEGORY	Properties containing the transaction totals such as transaction amount, balance due, change due, tax amount, subtotal, etc.
WorkstationStatusProperties	"WORKSTATION_STATUSES"	WorkstationStatusProperties.CATEGORY	Properties such as terminal number, transaction number, transaction in progress indicators, training mode indicator, offline indicator, etc.

Table 6 Data Provider Property Categories

The following figure illustrates the inheritance hierarchy of the Data Provider Property interfaces.

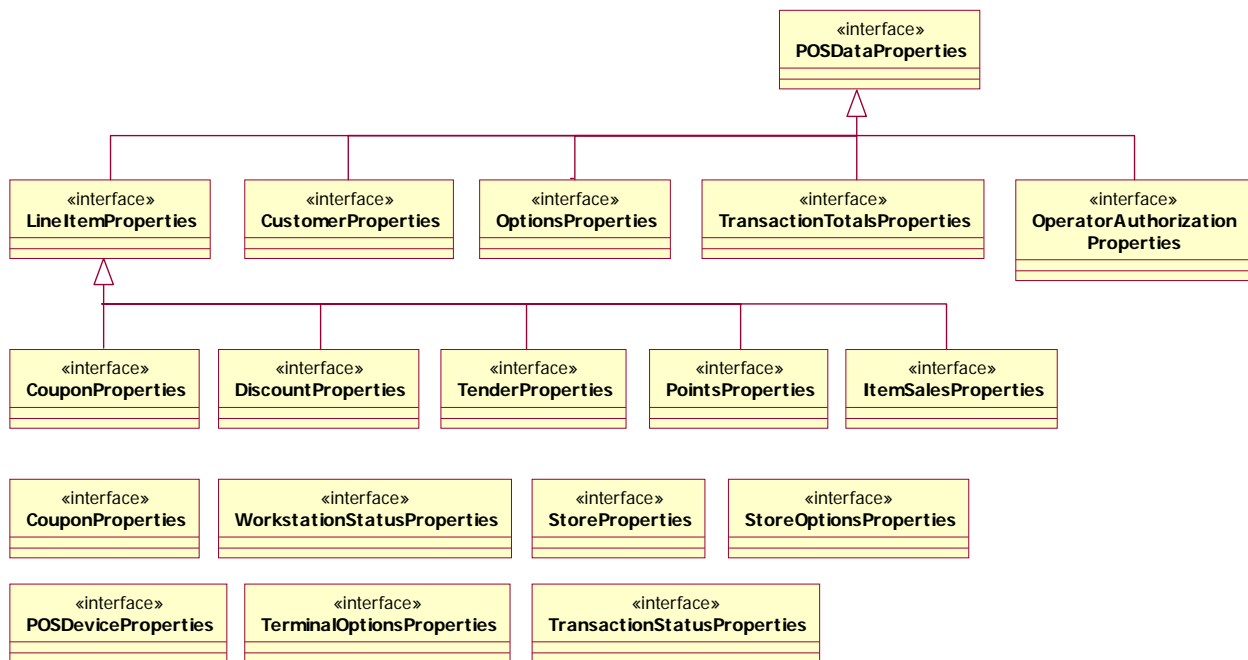


Figure 13 Data Provider Property Hierarchy

Event and Listener Types

The Data Provider supports a number of event and listener types. The following figure illustrates the event type hierarchy.

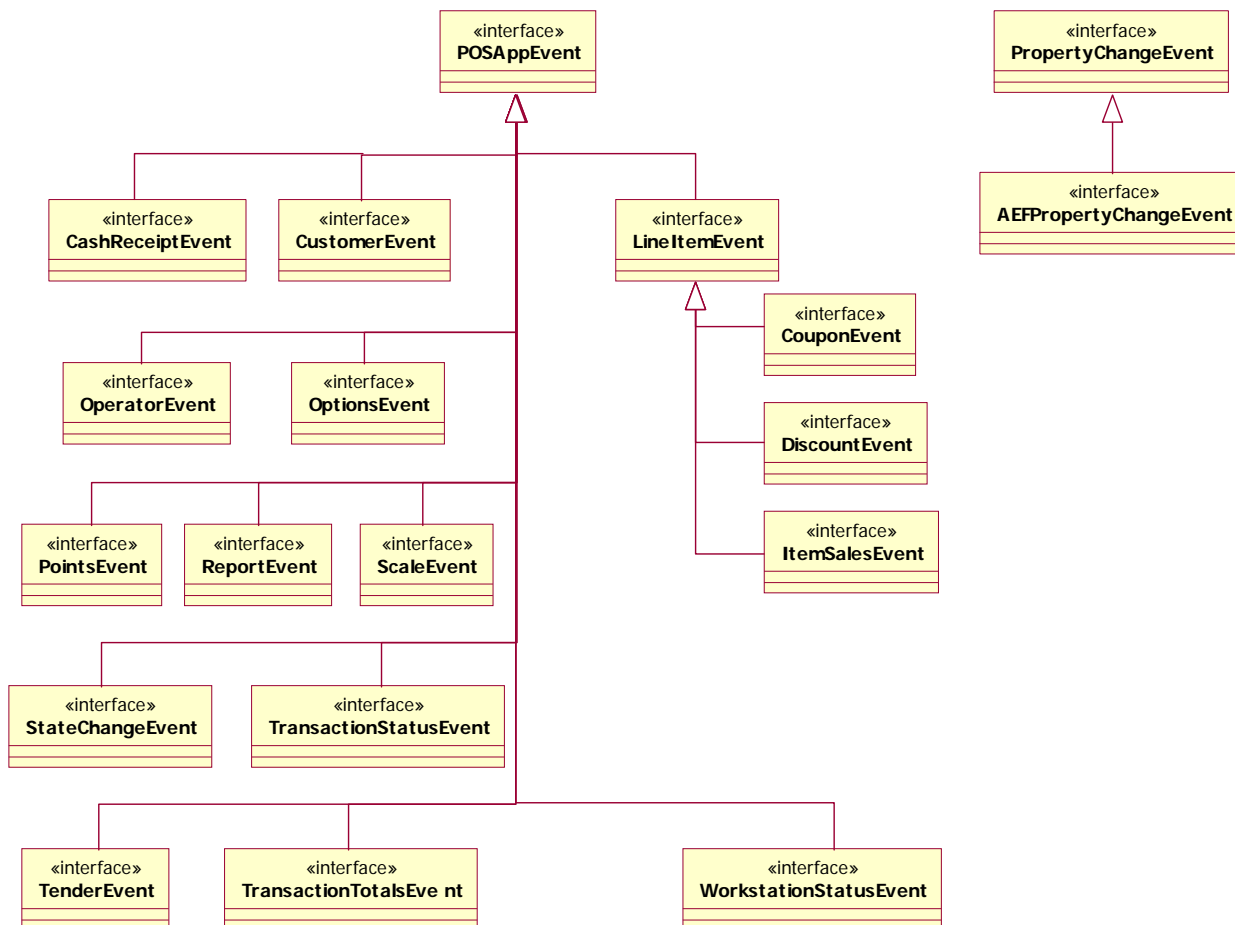


Figure 14 POSDataProvider Event Hierarchy

Each of the event types shown above is paired with a corresponding listener interface. The event and listener types are discussed below.

AEFPPropertyChangeEvent

The [AEFPPropertyChangeEvent](#) is fired when an individual Data Provider property changes value. The event includes the property category, property name, old value, and new value. See Appendix C. Data Provider Properties on page 280 for the properties supported by the Data Provider.

AEFPPropertyChangeListener

To receive AEFPPropertyChangeEvents, a client must implement the [AEFPPropertyChangeListener](#) interface, and register with the `POSDataProvider.addAEFPPropertyChangeListener` method.

CashReceiptEvent

The [CashReceiptEvent](#) is fired when the POS application sends print lines to the cash receipt printer.

CashReceiptListener

To receive CashReceiptEvents, a client must implement the [CashReceiptListener](#) interface, and register with the `POSDataProvider.addCashReceiptListener` method.

CouponEvent

The [CouponEvent](#) is fired whenever a store or manufacturer's coupon is added to or voided from the transaction. The event contains information such as the coupon amount, manufacturer number (for manufacturer's coupons), pricing method, etc.

CouponListener

To receive CouponEvents, a client must implement the [CouponListener](#) interface, and register with the `POSDataProvider.addCouponListener` method.

CustomerEvent

The [CustomerEvent](#) is fired when a customer loyalty number is added to the POS transaction. The event contains information such as the customer loyalty number, customer name, and points totals.

CustomerListener

To receive CustomerEvents, a client must implement the [CustomerListener](#) interface, and register with the `POSDataProvider.addCustomerListener` method.

DiscountEvent

The [DiscountEvent](#) is fired when a customer loyalty number is added to the POS transaction. The event contains information such as discount rate, discount amount, and an indicator as to whether the discount applies to the whole transaction.

DiscountListener

To receive DiscountEvents, a client must implement the [DiscountListener](#) interface, and register with the `POSDataProvider.addDiscountListener` method.

ItemSalesEvent

The [ItemSalesEvent](#) is fired whenever an item is added to or voided from the transaction. The event contains information such as the item code, pricing method, extended price, quantity, weight, foodstamp eligibility, etc.

ItemSalesListener

To receive ItemSalesEvents, a client must implement the [ItemSalesListener](#) interface, and register with the `POSDataProvider.addItemSalesListener` method.

LineItemEvent

The [LineItemEvent](#) is fired whenever a line item is added to or voided from a transaction. This is a base class for subclasses such as ItemSalesEvent, PointsEvent, and TenderEvent. The event contains information such as the line item description, a void indicator, a refund indicator, and print lines associated with the line item.

OperatorEvent

The [OperatorEvent](#) is fired when an operator signs on, signs off, or goes into secured mode. The event contains information such as the operator id, operator name, and an [OperatorAuthorization](#) object.

OperatorListener

To receive OperatorEvents, a client must implement the [OperatorListener](#) interface, and register with the `POSDataProvider.addOperatorListener` method.

OptionsEvent

The [OptionsEvent](#) is fired when the POS application loads a set of configuration (personalization) options. The event contains information such as a [StoreDefinition](#), [TerminalOptions](#), [StoreOptions](#), reason codes, etc.

OptionsListener

To receive OptionsEvents, a client must implement the [OptionsListener](#) interface, and register with the `POSDataProvider.addOptionsListener` method.

PointsEvent

The [PointsEvent](#) is fired when customer loyalty points are awarded, redeemed, or voided within a transaction. The event contains information such as the points category, the number of points, and an indicator whether the points were awarded or redeemed.

PointsListener

To receive PointsEvents, a client must implement the [PointsListener](#) interface, and register with the `POSDataProvider.addPointsListener` method.

POSAppEvent

The [POSAppEvent](#) is the base class for the Data Provider event interfaces. The event contains information such as the terminal number, event type, and property change category.

POSAppEventListener

The [POSAppEventListener](#) interface is a base type for the other listeners in this section.

ReportEvent

The [ReportEvent](#) is fired for specific reports generated from the register (such as the Open Transaction Report in ACE). The event contains information such as the report id, and a Collection of [ReportSection](#) instances.

ReportListener

To receive DiscountEvents, a client must implement the [ReportListener](#) interface, and register with the `POSDataProvider.addReportListener` method.

ScaleEvent

The [ScaleEvent](#) is fired when a the scale registers a change (item placed onto scale, or removed from scale). The event contains information such as weight, and unit of measure.

ScaleListener

To receive ScaleEvents, a client must implement the [ScaleListener](#) interface, and register with the `POSDataProvider.addScaleListener` method.

StateChangeEvent

The [StateChangeEvent](#) is fired when the POS application transitions between states. For 4690 POS applications, the state is a number corresponding to the application's state table as processed by the I/O processor. The event contains the new state number, and an indicator if the state transition was a "change to current" state as defined in the state table.

StateChangeListener

To receive DiscountEvents, a client must implement the [StateChangeListener](#) interface, and register with the `POSDataProvider.addStateChangeListener` method.

TenderEvent

The [TenderEvent](#) is fired whenever a tender line item is added to or voided from a transaction. The event contains information such as the tender type and variety, the tender amount, fee, and account number.

TenderListener

To receive TenderEvents, a client must implement the [TenderListener](#) interface, and register with the `POSDataProvider.addTenderListener` method.

TransactionStatusEvent

The [TransactionStatusEvent](#) is fired when there is a specific change in the state of the POS transaction. The event contains information such as transaction id, date and time, category, type, indicators whether discounts are allowed, tax change information, etc.

TransactionStatusListener

To receive TransactionStatusEvents, a client must implement the [TransactionStatusListener](#) interface, and register with the `POSDataProvider.addDiscountListener` method.

TransactionTotalsEvent

The [TransactionTotalsEvent](#) is fired whenever any running totals within a transaction are updated. The event contains information such as transaction total amount, subtotal amount, tax amount, balance due, change due, food stamps total, food stamps balance due, food stamps change due, total items, total coupons, coupon amount, etc.

TransactionTotalsListener

To receive TransactionTotalsEvents, a client must implement the [TransactionTotalsListener](#) interface, and register with the `POSDataProvider.addTransactionTotalsListener` method.

WorkstationStatusEvent

The [WorkstationStatusEvent](#) is fired whenever there is a specific change in the POS register. The event contains information such as indicators that the input sequence was cleared, or that the terminal acknowledged that it has been disabled.

WorkstationStatusListener

To receive WorkstationStatusEvents, a client must implement the [WorkstationStatusListener](#) interface, and register with the `POSDataProvider.addWorkstationStatusListener` method.

Extending POSAppEvents

This section provides a general overview of [POSAppEvent](#) objects and provides an introduction to their usage. It also provides information needed to utilize the extension mechanism provided by [POSAppEventElement](#) which is the parent class of [POSAppEvent](#). [POSAppEventElement](#) provides a generalized “event element” container for additional event attributes and sub-elements. This mechanism can be used to add user data into the XML messages sent from the POS application, and have them automatically included in the relevant [POSAppEvent](#) automatically.

[POSAppEvents](#) are created by parsing the XML messages provided from the base POS application. For example, a [CustomerEvent](#) object is the result of parsing XML defined by the `<customer>` XML message (see “Customer Event” on page 318). The type of event object to be created is defined by the following line in the `classes.properties` file:

```
customer=com.ibm.retail.AEF.event.CustomerEventImpl
```

By default, the XML element tag is used as a key to lookup the event class in `classes.properties`. If no entry is found, then the [POSAppEventImpl](#) class is used.

The [CustomerEventImpl](#) class is a subclass of [POSAppEventImpl](#) and implements an API unique for customer data (e.g., `customer.getID` returns the customer loyalty number). While the [CustomerEvent](#) interface provides a set of methods which cover the basics of customer data, there may be cases where additional data is desired. In this case the more general interface provided by [POSAppEventElement](#) may be used.

For example, suppose a new attribute “foo” is needed in the customer event. The POS application can modify the XML sent to include this attribute in the customer XML:

```
<customer id="90071234566" foo="820515" />
```

Note: See “Task: Adding Additional Data to an Event” on page 74 for information on how to modify the XML messages sent to the AEF by the POS application.

Although no method for “`getFoo()`” is provided in the [CustomerEvent](#) interface, the value of the attribute `foo` is available by using the `getProperty` method of the [POSAppEventElement](#) interface inherited by [CustomerEvent](#). This approach is illustrated below:

```
public void customerCardEntered(CustomerEvent evt) throws
RemoteException {
    String foo = (String) (evt.getProperty("foo"));
    if (foo != null) {
        System.err.println("The value of foo is " + foo);
    } else {
        System.err.println("No foo in this event!");
    }
}
```

If `foo` is a complex element rather than a simple attribute, a `<foo>` element may be included in the XML. In this case, the AEF will attempt to create an object of the appropriate type by referencing the `userclasses.properties` file for the correct classname. The class must be a subclass of the [POSAppEventElementImpl](#) class. If no entry is found, then an object of type [POSAppEventElement](#) is created by default. For example, if the XML sent from the POS application is:

```
<customer id="90071234566">
```

```
<foo name="myfoo" attribute1="some more data" isFoo="true" />
</customer>
```

The following code would provide access to the attributes of the <foo> element:

```
public void customerCardEntered(CustomerEvent evt) throws
RemoteException {
    Object element =
    (POSAppEventElement) (evt.getProperty("foo"));
    if (element != null) {
        if (element instanceof Collection) {
            System.err.println("We have many foos");
        }
    } else {
        POSAppEventElement foo = (POSAppEventElement)element;
        System.err.println("The value of foo attribute1 is " +
        foo.getProperty("attribute1"));
    }
}
```

Listener Considerations

The POSDataProvider maintains a queue of events that must be dispatched to listeners. The listener event notification mechanism is synchronous. This is of special concern for clients of virtual sessions. If a client delays or blocks the POSDataProvider dispatch thread while processing an event, then event dispatching for all virtual sessions will be delayed or blocked until the client returns from handling its event.

When handling a data provider event, the client listener should process the event as quickly as possible. If the processing is not trivial, the client should handle the event on another thread instead of the data provider dispatch thread.

In order to make efficient use of the data provider dispatch thread, the AEF includes proxy listener classes which may be used by the clients. The proxy performs event queuing and thread context switching so that the events are automatically handled on a different thread than the data provider event dispatch thread.

It is recommended that clients use the proxy listeners when possible, especially when the client is remote.

For each of the data provider event listener classes, there is a corresponding proxy class as listed in the table below.

"Regular" Listener Class	Proxy Listener Class
CashReceiptListener	CashReceiptListenerProxy
CouponListener	CouponListenerProxy
CustomerListener	CustomerListenerProxy
DiscountListener	DiscountListenerProxy
POSAppEventListener	GenericListenerProxy
ItemSalesListener	ItemSalesListenerProxy
OperatorListener	OperatorListenerProxy
OptionsListener	OptionsListenerProxy
PointsListener	PointsListenerProxy
ReportListener	ReportListenerProxy

ScaleListener	ScaleListenerProxy
SessionStatusListener	SessionStatusListenerProxy
StateChangeListener	StateChangeListenerProxy
TenderListener	TenderListenerProxy
TransactionStatusListener	TransactionStatusListenerProxy
TransactionTotalsListener	TransactionTotalsListenerProxy
WorkstationStatusListener	WorkstationStatusListenerProxy

Table 7 Data Provider Event Listener Proxies

See “Task: Subscribing to a Data Provider Event” on page 72 for an example of subscribing to a Data Provider event.

Systems Management

The systems management implementation employed within the Store Integrator is based entirely on JMX, which forms the basis of the management infrastructure for WebSphere. JMX is defined by JSR-003, and deals only with management issues within the context of a single JVM. Each management agent running within the Store Integrator environment is built upon a single JMX agent. In order to complete the Store Integrator's hierarchical management infrastructure, some additional remote connectivity support is required. This support provides the communications bridge between each of the management roles previously discussed. The standard that defines the remote interfaces for accessing a JMX Agent, JSR-160, provides this remote connectivity support.

Components

As noted earlier, there are two primary components that comprise the JMX infrastructure for the Store Integrator environment: The General Agent, and the Master Agent. These roles are distinguished by virtue of the MBeans that are attached to them, with the General Agent receiving MBeans designed for the client, and the Master receiving those defined for the controlling role. In a single JVM, there cannot be more than one management agent running. This is controlled by the [MgmtAgentFactory](#) class, which provides access to a [MgmtAgent](#) instance within the current JVM.

General Agent

The General Agent is designed for use within all endpoint devices within the store network. Examples of which include the POS controllers and terminals, WebPads, and Kiosks. The General Agent will, by default, include MBeans to handle client discovery, JVM health information, and configuration and forwarding for logging.

Master Agent

The Master Agent is based on the same agent core as the General Agent, with the following exception. The Master Agent is the single point of access into the Store Integrator environment by any management agent or application, and it has a different set of MBeans loaded at agent startup. This set of MBeans includes the JVM Health and logging MBeans loaded by the General Agent, plus MBeans for managing the proxy relationships with General Agents running within the store. Also included is an MBean for aggregating the notifications (events) generated by each General Agent. Each of these components will be discussed further in the following sections.

Component Relationships

The role of the Master Agent is to discover the General Agents that exist within the environment, to then create proxy MBeans for MBeans that exist on those General Agents, and to control all interactions between them. By doing this, all of the MBeans in the environment are concentrated at the Master Agent, thus giving a management application one (and only one) point of control for the entire environment. This functional hierarchy becomes even more important for disciplines that require some distributed policy control, such as Software Distribution or Device Monitoring. In each of these cases the control point, or the point where a management application interacts with the Store Integration Framework, is *only* the Master Agent. Components responsible for managing a discipline on the Master Agent will then interact in turn with the appropriate General Agent MBeans to carry out an operation.

A good example of this would be the application of a monitor. A management application may wish to have a device monitor created and applied on a set of WebPad(s). The operation is actually carried out by creating and registering monitor policy on the Master Agent's Monitor Policy Management MBean. As each of the target device(s) (in this case all WebPads) become active, monitors will be applied to the device. But if the device is offline, or if a current one goes off line and then comes back, the monitor policy MBean will be responsible for reapplying the monitor, based on the discovery of a new WebPad device. This gives the entire Store Integration Framework environment a certain amount of independence from any and all management applications. An application can attach to the Store Integration Framework Master Agent, setup some policy, and detach. Meanwhile, the environment will continue to function autonomously based on the established policy.

A quick look at how this fits in the practical sense within the Store Integrator environment, i.e., where all of the pieces are physically located in the implementation, is depicted in Figure 15 below, and described in detail in the ensuing sections.

The role of the Master Agent is to discover the General Agents that exist within the environment, to then create proxy MBeans for MBeans that exist on those General Agents, and to control all interactions between them. By doing this, all of the MBeans in the environment are concentrated at the Master Agent, thus giving a management application one (and only one) point of control for the entire environment. This functional hierarchy becomes even more important for disciplines that require some distributed policy control. In each of these cases the control point, or the point where a management application interacts with the Store Integration Framework, is *only* the Master Agent. Components responsible for managing a discipline on the Master Agent will then interact in turn with the appropriate General Agent MBeans to carry out an operation.

A quick look at how this fits in the practical sense within the Store Integrator environment, i.e., where all of the pieces are physically located in the implementation, is depicted in Figure 15 below, and described in detail in the ensuing sections.

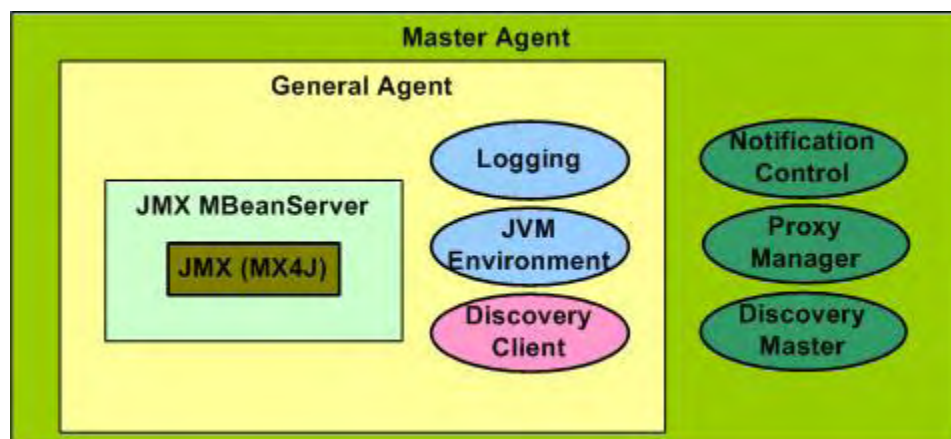


Figure 15 Management Infrastructure

Agent Discovery and Health Checking

This component provides a means for discovering management agents within the store, along with monitoring their continued health and status. The intent of this function is to provide a simple discovery and health checking mechanism that is both reliable and

inexpensive in the store environment. Subsequently, this function will trigger the creation and deletion of device information and Proxy MBeans on the Master Agent.

The implementation of this function requires two MBean interfaces, [MgmtClientHealthMBean](#), and [MgmtMasterHealthMBean](#). The [MgmtClientHealthMBean](#) interface, implemented as part of the General Agent, provides a simple periodic heartbeat and agent information. The [MgmtMasterHealthMBean](#) interface, implemented as part of the Master Agent, listens for client heartbeats, emits notifications when agents are discovered or go offline, and maintains information about each discovered agent. It also registers to listen for [JMXConnectionNotifications](#) from the JSR-160 implementation.

The primary reason for two separate health checks is that it affords the ability to check on the health of a device at two very different levels. The assumption is that in order for the JMX infrastructure to function normally, the box has to be very healthy, but for a low level heartbeat frame to flow, a considerably smaller amount of the system needs to be healthy and functional.

MgmtMasterHealthMBean

The [MgmtMasterHealthMBean](#) is located at one place in the store: the Master Agent. In the case of a full Store Integration Framework deployment, this is on the in-store processor. This component is responsible for:

- Listening for periodic heartbeats from agents running within the store
- Maintaining information about each discovered agent
- Controlling the emission of both [AgentDiscoveredNotifications](#) and [AgentLostNotifications](#), which trigger the construction and tearing down of connections and proxy MBeans between management agents

MgmtClientHealthMBean

This entity is located within each General Agent running within the store, and is responsible for:

- Collecting agent information for use in the agent discovery protocol
- Issuing periodic (based on a configured value) discovery frames for consumption by the [MgmtMasterHealthMBean](#)

The communications mechanism used by the discovery protocol is a single simple Multicast UDP PDU that is issued by the [MgmtClientHealthMBean](#) at regular, configurable intervals. UDP is chosen for simplicity, and for the minimum of dependencies on a healthy machine. It also makes the receiver logic far simpler, and requires extremely low bandwidth.

The [MgmtMasterHealthMBean](#) uses an algorithm such that the arrival of a discovery frame from an agent indicates it is available and causes a missed interval counter to be reset. If a configured number of discovery frames are missed, based on the interval configured for their transmission, then the agent is considered off-line. Bear in mind that this exchange of frames is not meant to be a robust form of health checking, but rather a general purpose way of indicating the monitored status of the agents in the store. Since this mechanism is based on multicast UDP, this multicast traffic is only required within

the store, and in fact should be limited to the store only. The protocol uses the following multicast address and port assignments:

- **Multicast Address:** 225.6.29.63
- **Port:** 31200

Notification Control

The responsibility of this function is to consolidate the Notifications issued by all of the General Agents in the Store Integrator environment at the Master Agent, and to provide them to users of the management API when needed. Since management agents and applications may not always be connected to receive notifications in real time, this component manages a given amount of notifications on a persistent and temporary basis, and provides filtering for those notifications that are persisted. The MBean that provides the implementation for this component, the [MgmtNotificationControlMBean](#), is part of the base set instantiated on the Master Agent. Multiple instances can be created for different levels of filtering.

There are three entities are involved in this function:

Notification Generation

For an MBean to issue a Notification into the system, nothing beyond the base Notification forwarding mechanisms provided by JMX is required.

Notification Collection and Control

The job of this entity is to register as a listener for all notifications generated by each General Agent. These listeners are registered as a result of the proxy relationship established between the Master and General Agents. Upon receiving each notification, this entity will re-emit it locally and will hand it over to the Persistent Notification Store. As a result of the notifications being re-emitted, a management application or management agent can add a `NotificationListener` to the `MgmtNotificationControlMBean` itself and receive all generated notifications.

Persistent Notification Store

The Persistent Notification Store is responsible for maintaining in a persistent, reboot-safe manner, the stream of received Notifications. The store houses a configurable number of notifications in a circular queue, and persists only those notifications that pass through a filter based on notification class name. The `MgmtNotificationControlMBean` defines an interface perusing the stored notifications, allowing a management application or agent to grab a past snapshot of system-wide notifications that may have been emitted when the application was not active. Figure 16 illustrates how all four pieces fit together to provide a unified notification mechanism that allows for distributed filtering, with centralized collection, control and logging.

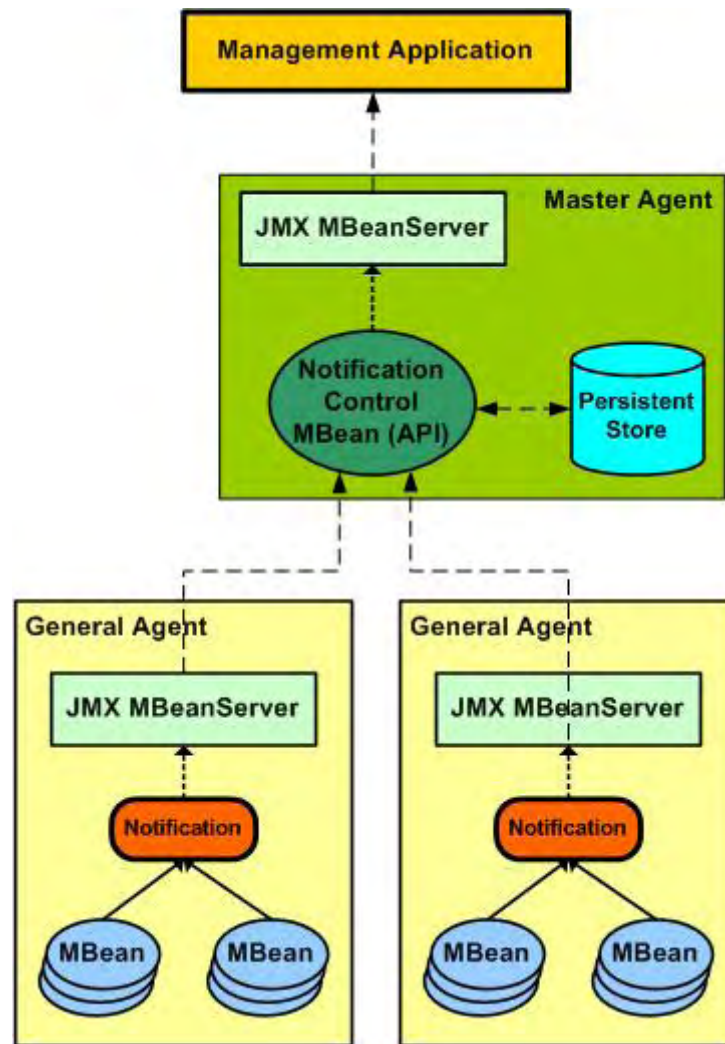


Figure 16 Notification Control Mechanism

Logging Configuration and Forwarding

The responsibility of this component is twofold: To provide the ability to remotely make non persisted changes to logging levels, and to enable or disable the forwarding of log events as JMX Notifications. These functions are handled by separate MBeans created automatically by the Store Integrator management agents. Remote log forwarding is supported for JDK 1.4 logging, Log4J, and Syslog, while logging level changes are supported for JDK 1.4 logging and Log4J.

The [MgmtLoggingCtrlMBean](#) controls the creation of all logging MBeans during agent startup. Based on the logging packages detected, the appropriate logging and remote logging MBeans will be instantiated. The following is a summary of the logging MBeans that can get created:

MBean Name and ObjectName Id	Description
MgmtLoggingCtrlMBean	MBean started on all agents that sets up remote logging MBeans and internal logging MBeans for all applicable log types. The default remote logging level is OFF.
RemoteLog4JLoggingCtrlMBean (id=RemoteLog4JLogCtrl)	Sets up the forwarding of Log4J messages as RtlTracePointNotifications. Created if the system detects Log4J.
Log4JLoggingMBean (id=Log4JLoggers)	MBean that allows for dynamically changing the Log4J logging level (TRACE,DEBUG,etc) for any Logger in the hierarchy. Created if the system detects Log4J.
RemoteJDKLoggingCtrlMBean (id=RemoteJDKLogCtrl)	Sets up the forwarding of JDK messages as RtlTracePointNotifications. Created if the system detects JDK logging. Important: Set the pushInterval attribute to a high value (10-15) when forwarding TRACE and DEBUG messages.
JDKLoggerMBean (id=JDKLoggers)	MBean that allows for dynamically changing the JDK logging level (FINEST,FINE,etc) for any Logger in the hierarchy. Created if the system detects JDK Logging,
JDKHandlerMBean (id=JDKHandlers)	MBean for dynamically changing the JDK logging level for any of the JDK Logging Handlers
RemoteSyslogLoggingCtrlMBean (id=RemoteSysLogCtrl)	Sets up the forwarding of local Syslog messages as RtlTracePointNotifications. Created for device types dTypeROLO and dTypeConsumer. There is no dynamic control of internal logging like with Log4J and JDK. The intention is to have one MBean running per physical device running syslogd.

Table 8 Logging MBeans

The RemoteLoggingCtrlMBean has its own set of logging levels that map to each logging implementation. These levels are used to determine which log messages get forwarded as notifications. The following table lists the RemoteLoggingCtrlMBean's logging levels, and how they map to the levels of each logging implementation:

RemoteLoggingCtrlMBean	JDK 1.4 Logging	Log4J 1.2.8	Syslog
LEVEL_EMERGENCY (1)	Level.SEVERE	Level.FATAL	SYSL_LOG_EMERG(0)
LEVEL_ALERT (2)	Level.SEVERE	Level.FATAL	SYSL_LOG_ALERT(1)
LEVEL_CRIT (4)	Level.SEVERE	Level.FATAL	SYSL_LOG_CRIT(2)
LEVEL_ERR (8)	Level.WARNING	Level.ERROR	SYSL_LOG_ERR(3)
LEVEL_WARNING (16)	Level.WARNING	Level.WARN	SYSL_LOG_WARNING(4)
LEVEL_NOTICE (32)	Level.INFO	Level.INFO	SYSL_LOG_NOTICE(5)
LEVEL_INFO (64)	Level.INFO	Level.INFO	SYSL_LOG_INFO(6)
LEVEL_DEBUG (128)	Level.FINEST	Level.DEBUG	SYSL_LOG_DEBUG(7)

LEVEL_MASK_OFF (0)	Level.OFF	Level.OFF	<Off>
LEVEL_MASK_SEVERE (7)	Level.SEVERE	Level.FATAL	SYSL_LOG_CRIT(2)
LEVEL_MASK_ALL	Level.ALL	Level.ALL	SYSL_LOG_DEBUG(7)

Table 9 RemoteLoggingCtrlMBean Log Levels

Inventory

Inventory as a management discipline is as much an application statement as one of enablement. That is, the function of inventory is to collect data about both hardware and software components that comprise the system as a whole. In reality, it is little more than the collection of specific attribute information from the agent infrastructure. As such, the function of collecting inventory is different than the discussion of providing the correct information for collection. The process of collection is best left to the management application vendors or as part of the design of the Store Integrator Viewer. However, the requirements to define a base level of inventory to be made available to those applications is dealt with below.

Given that each software and hardware component can be represented by MBeans, it seems that the most reasonable approach for introducing inventory data is to define Java Interfaces that support the retrieval of this information. That way the inventory collection can be added to the MBean that the developer/designer of the component feels it is most appropriate. The consumer of the inventory data, namely management agents and applications, need only search out the predefined interfaces using normal JMX interactions. The Store Integrator management API defines two MBean interfaces for software and hardware inventory, [MgmtSoftwareInventoryMBean](#) and [MgmtHardwareInventoryMBean](#). These interface definitions are based on the work done by the DMTF for both DMI, and CIM. This information is consistent with that used by the Tivoli management software as well.

Store Integrator Viewer

The Store Integrator Viewer is a simple to use configuration and management tool for use either in store or from the enterprise that can handle certain base-line configuration, event display, and other management functions for a single store. This tool is intended to fill the gap for those customers that have not yet deployed a management solution in their environment. To that end, it is not intended to be a substitute for the function provided by Tivoli or any other third party vendor.

The design of this tool is as a J2EE Web application that runs on WebSphere Application Server. The tool provides the following functions:

- Support of a *single* store only
- Views for each device in the store
- Inventory
- Notification (Event) log viewer with filters

The base application is a simple MBean Browser that interacts with the Master Agent, capable of interacting with any and all MBeans that are discovered by the agent infrastructure. Using the browser, one can:

- View and set MBean attributes involving primitive Java types
- Invoke MBean operations involving primitive Java types
- View and filter JMX Notifications emitted by General Agents within the store

Figure 17 depicts the Store Integrator Viewer interface. The MBeans are displayed in a hierarchical manner, based first on device, then by component (AEF, AEFIO, etc), and finally by identifier (GUISoftwareInventory).

The screenshot shows the IBM Store Integrator Viewer interface. The top navigation bar includes 'Home' and 'logout (testuser)'. The left sidebar shows a hierarchical tree of connected devices under 'Store 88', including 'Device IS.001.11099' and 'Device IR.10151'. The main content area displays the 'GUISoftwareInventory' MBean, which is a 'Manageable Bean'. Below this, the 'Properties' tab is active, showing a table of 'General Properties'.

Property Name	Property Value	Description
BuildNumber	0225	Attribute exposed for management
CurrentState	7	Attribute exposed for management
Description	Store Integrator Graphical User Interface	Attribute exposed for management
FixLevel	0225	Attribute exposed for management
InstallationDate		Attribute exposed for management
MajorVersion	1	Attribute exposed for management
Manufacturer	IBM	Attribute exposed for management
MinorVersion	0	Attribute exposed for management
ProductName	Store Integrator Graphical User Interface	Attribute exposed for management
SerialNumber		Attribute exposed for management

At the bottom, the 'System Overview' section provides a summary of notifications and connectivity. It shows 62 notifications (red circle), 96 warnings (yellow triangle), 99 errors (green square), and 94 events (star). A 'View All 351' link is available. The 'Connectivity' section shows 3 devices and 6 components.

Figure 17 Store Integrator Viewer

Chapter 3. Programming with the AEF

Getting Started

Task: Setting up the Development Environment

In order to modify AEF configuration properties, or extend the AEF, you must setup a Java development environment. At a minimum you will need a Java Development Kit (JDK), a text editor, and some additional zip and jar files. For the instructions which follow, we assume that the development platform is Windows based, however, any platform with the appropriate JDK will be acceptable.

Java Development Kit

The Java Development Kit can be downloaded from the internet for free. You should download version 1.4 or higher. Then follow the instructions to install it into your development system.

Additional Files

In order to create applications that use Store Integrator functionality, the Java compiler needs access to Store Integrator and 4690 specific classes. The following files should be copied in a directory on your development system called C:\si\si_jars\.

- Files from the 4690\COM directory on Store Integrator CD:
 - siutil.jar
 - simgmt.jar
 - c_logging.jar
- Files from 4690\AEF directory on Store Integrator CD:
 - aef.jar
 - aefio.jar
- Files from C:\java\lib directory on the 4690 controller:
 - os4690.zip
- Files from C:\java directory on the 4690 controller:
 - jpos4690.zip
 - poss4690.zip
 - jpos14.jar
 - ibmjpos.jar
 - jattach.jar
 - tss.jar
 - log.jar
 - comm4690.jar
- Files from <http://mx4j.sourceforge.net> website:
 - mx4j_rmt.jar
 - mx4j.jar

User Hierarchy

The examples in this guide assume a specific directory structure. The following directories should be created on the development machine.

Directory	Purpose
d:/si	Base Store Integrator development directory.
d:/si/si_jars	Contains Jar and Zip files required for building Store Integrator extensions.
d:/si/user	Root directory for user Java source code.
d:/si/userjar	Root directory for user property file overrides and user Java class files.

Table 10 Development Machine Directory Hierarchy

Task: Using the Quick Start Sample

This document is packaged with precompiled source for a sample AEF client program which demonstrates various aspects of the AEF API. The following are the prerequisites for running the sample client.

1. The development environment is installed on the client as described on page 63.
2. The POS application has been AEF enabled as required on the 4690 system.
Note: There is an “AEF Application Emulator” (also known as “possim”) which provides limited POS application functionality without requiring an AEF enabled application to be running on a 4690. This tool is strictly an “as-is” tool with neither support nor warranty of any type implied. The sample client can be run against this emulator without requiring either a 4690 machine, or an AEF enabled application. This may be of use to get the client running quickly. To get the “AEF Application Emulator”, contact your IBM Retail Store Systems Technical Marketing representative.
3. CSS is configured and running one or more virtual sessions.

In this task, you will execute the “SampelGUI” java client which will connect to a `SessionServer`, claim an `AEFSession`, and display a simple GUI which allows you to interact with the session.

Running the SampleGUI Client

To run the SampleGUI on a Windows client, execute the command:

```
>java -Djava.rmi.server.hostname=xx.xx.xx.xx -classpath  
d:/si/user;d:/si/si_jars/aef.jar;d:/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar;d:/si/si_jars/simgmt.jar  
com.userco.retail.client.gui.SampleGUI
```

where **xx.xx.xx.xx** is the TCP/IP address of the network interface card on the client machine which is connected to the machine running the virtual sessions, and **d** is the drive letter where the zipfile containing this document was extracted.

If the client is setup correctly, you will see the following progress screen which is displayed while the session is retrieved and initialized.

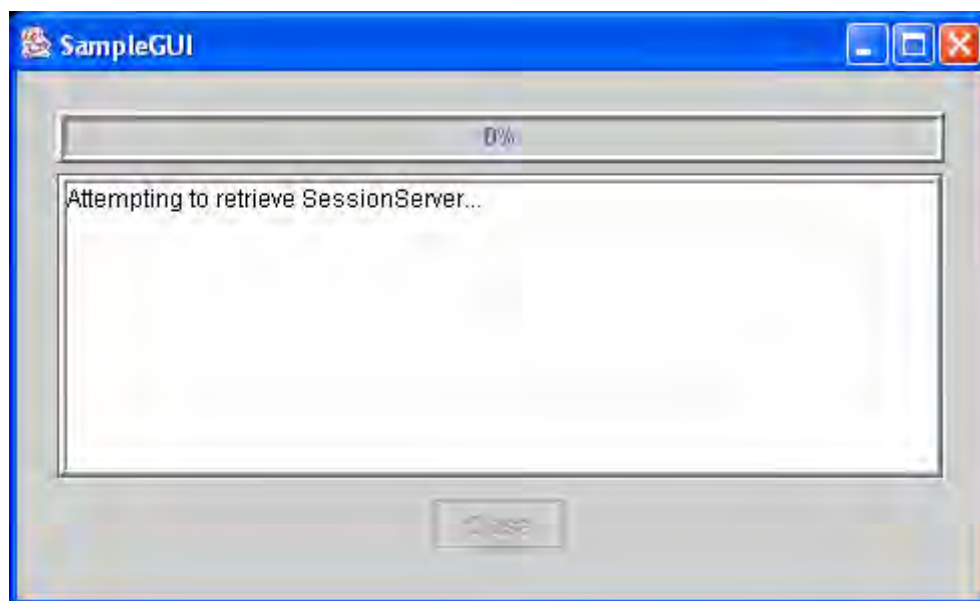


Figure 18 SampleGUI Initial Progress Screen

Once the session has been accessed and initialized, the following screen is displayed.

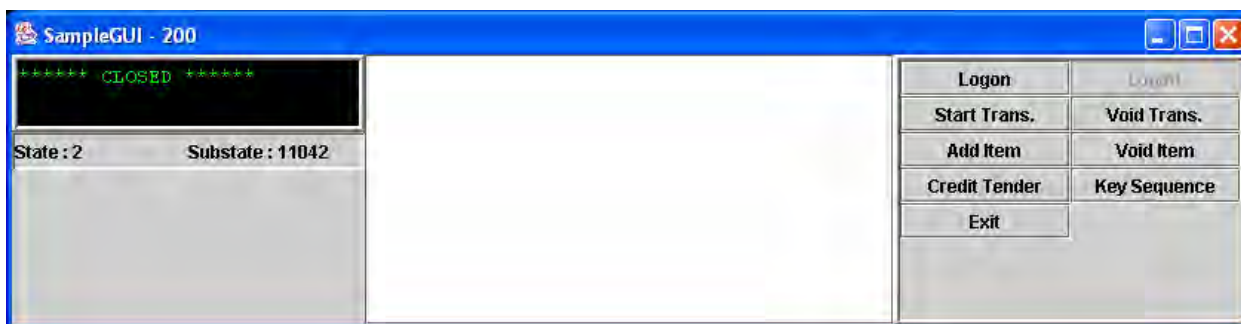


Figure 19 SampleGUI Main Screen

The main screen includes an operator display, a status area showing the POS application state and substate, a cash receipt area, and a button palette which allows interaction with the POSAutomationProvider API.

AEF

Infrastructure

Task: Getting a SessionServer

In order to attach to a virtual or real session, it is necessary to find an instance of the `SessionServer`. The simplest way to get a `SessionServer` is to use the `AEFBase.getInstance().getSessionServer()` method. This method causes a broadcast beacon to be sent out, and the first `SessionServer` instance that responds will be used. This will only locate `SessionServer` instances within the same TCP/IP subnet as the client. The next task describes connecting to a `SessionServer` outside the subnet.

In this task, you will use the simple method of finding a `SessionServer`. The steps are

- Write the `AEFExample1` class, including the call to the `getSessionServer` method.
- Compile the class.
- Execute the client code, assuming that CSS is running with virtual sessions on a 4690 within the same subnet as the client.

Writing the AEFExample1 Class

A complete `AEFExample1` class has been provided in `d:/si/user/com/userco/retail/client/example1/AEFExample1.java`. Examine the source code and find the call to the `getSessionServer` method.

Compiling the Code

The code has already been compiled, but if you want to modify it, you can recompile it with the command:

```
d:\si\user>javac -classpath
d:/si/si_jars/aef.jar;d:/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar
com/userco/retail/client/example1/AEFExample1.java
```

Executing the Code

To execute the code, execute the command:

```
d:\si\user> java -Djava.rmi.server.hostname=xx.xx.xx.xx -classpath .;d:/si/si_j
ars/aef.jar;d:/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar;d:/si_jars/simgm
t.jar com.userco.retail.client.example1.AEFExample1
```

Where `xx.xx.xx.xx` is the TCP/IP address of the client network interface card which is on the same network as the `SessionServer`.

Task: Connecting to a Specific SessionServer

This task demonstrates how to connect to a specific SessionServer, or a SessionServer which is outside of the client's TCP/IP subnet. To do this, you must:

- Determine the URI (Uniform Resource Identifier) of the SessionServer.
- Use the `AEFBase.getRemoteServerFromURI` method to fetch the SessionServer.

Determining the SessionServer URI

The SessionServer URI will be of the form:

```
rmi://xx.xx.xx.xx:yyyyy/server_name
```

where

xx.xx.xx.xx is the TCP/IP address of the machine where the SessionServer is running

yyyyy is the rmi port number for the JVM running the SessionServer. This is configured in the config.properties bundle under the keys "rmi.port", "4690.controller.rmi.port", and "general.rmi.port". The default port for the 4690 controller is 12099. The default port for all other platforms is 11099.

server_name is the id of the SessionServer as configured in the config.properties bundle. The value is configured under the keys "server.id" and default to `AEF_SESSION_SERVERnodename`.

Using the getRemoteServerFromURI Method

Assuming that the SessionServer URI is "rmi://10.30.151.1:12099/AEF_SESSION_SERVERDD", then the following line may be used to attempt to connect:

```
AEFBase.getInstance().getRemoteServerFromURI("rmi://10.30.151.1:12099/AEF_SESSION_SERVERDD");
```

Task: Getting a Specific Session

Clients (such as the remote SI GUI) may need to connect to a specific session. This is achieved via one of the `SessionServer.getSession` methods. If the session already exists, it will be returned to the caller. If the session doesn't already exist, but can be created, it will be returned to the client after it is created.

Assuming that the session server has been retrieved, the following lines will attempt to fetch an AEFSession corresponding to session number 201.

```
AEFSession session = null;
```

```
try
{
    session = sessionServer.getSession("201");
```

```

    }
    catch (Throwable th)
    {
        System.error.println("Unable to retrieve requested session, " + th.getMessage());
    }
}

```

Because session initialization can take some time, there may be a delay between the time the session is created, and the time the session is ready for use. By default, the `getSession` method will block until the requested session is ready for use, or until a configurable timeout occurs. This timeout value is specified in the "config.properties" bundle via the "ready.wait.timeout" key.

Note that retrieving a specific session does not stop other clients from specifically retrieving the same session. Another client could request session 201, even though the first client has already retrieved session 201.

Task: Getting a Session Without a Ready Wait

As discussed in the previous task, the default versions of the `SessionServer.getSession` and `SessionServer.getAvailableSession` methods block the caller until the session is ready for use, or until a timeout occurs. As an alternative, the client may wish the session to be returned as soon as it is created. The client can register as a `SessionStatusListener` so that it can be notified when the session becomes ready.

This task will demonstrate the use of a [SessionParameters](#) argument to indicate that the session should be returned immediately. It will also demonstrate the use of a [SessionStatusListenerProxy](#) which provides an event indicating when the session is ready.

Examine the `getAvailableSession` method from the `d:/si/user/com/userco/retail/client/gui/SampleGUI.java` module:

```

1.  /**
2.   * Retrieves an available session, and does not wait for it to become ready.
3.   *
4.   * @return AEFSession
5.   *
6.   */
7.  public static AEFSession getAvailableSession()
8.  {
9.      AEFSession retSession = null;
10.     SessionParameters parms = new SessionParameters();
11.     // A "-1" timeout value indicates that there should be no wait
12.     // for the session to become ready. It should be returned immediately.
13.     parms.setTimeout(-1);
14.     progressFrame.appendText("Requesting available session...");
15.     try
16.     {
17.         retSession = server.getAvailableSession(parms);
18.         progressFrame.setBarValue(20);
19.         try
20.         {
21.             progressFrame.appendText("Success, ID=" +
retSession.getTerminalNumber() + ".\n");
22.         }
23.         catch (RemoteException re)
24.         {

```

```

25.         log.error("Failed to retrieve AEFSession ID.", re);
26.     }
27.
28.     // Attach a SessionStatusListener to update the progress frame.
29.     try
30.     {
31.         listenerProxy = new
SessionStatusListenerProxy(retSession,
32.                             new SessionStatusListener()
33.                             {
34.                                 /**
35.                                  * Indicates a session status event has occurred.
36.                                  *
37.                                  * @param evt contains details of SessionStatusEvent
38.                                  * @throws RemoteException
39.                                  */
40.                                 */
41.                                 public void sessionStatusChanged(SessionStatusEvent evt)
42.                                 {
43.                                     progressFrame.appendText(evt.getDescription() + "\n");
44.                                     progressFrame.setBarValue( 20 +
45.                                     evt.getPercentComplete());
46.                                 }
47.                                 /**
48.                                  * Indicates that the session is ready for commands.
49.                                  *
50.                                  * @param evt contains details of SessionStatusEvent
51.                                  * @throws RemoteException
52.                                  */
53.                                  */
54.                                  public void sessionReady(SessionStatusEvent evt)
55.                                  {
56.                                      progressFrame.appendText(evt.getDescription() + "\n");
57.                                      progressFrame.setBarValue( 20 +
58.                                      evt.getPercentComplete());
59.                                      try
60.                                      {
61.                                          listenerProxy.removeListener();
62.                                      }
63.                                      catch (Throwable th2)
64.                                      {
65.                                          log.error("Unable to remove SessionStatusListener.",
66.                                          th2);
67.                                      }
68.                                  }
69.                                  /**
70.                                   * Indicates that the session has ended.
71.                                   *
72.                                   * @param evt contains details of SessionStatusEvent
73.                                   * @throws RemoteException
74.                                   */
75.                                   */
76.                                   public void sessionEnded(SessionStatusEvent evt)
77.                                   {
78.                                   }
79.                               }
80.
81.         // If the session is already ready, we can remove the
82.         // session status listener, and update the progress frame.
83.         if (retSession.isReady())
84.         {
85.             progressFrame.appendText("Session is ready.\n");
86.             progressFrame.setBarValue( 120 );
87.             listenerProxy.removeListener();
88.         }
89.     } catch (Throwable th)
90.     {

```

```

91.         log.error("Unable to add SessionStausListener.", th);
92.     }
93. }
94. catch (Throwable e)
95. {
96.     progressFrame.appendText("Failed.\n");
97.     progressFrame.appendText("    Exception : " + e.toString() + "\n");
98.     progressFrame.setButtonEnabled(true);
99.     log.error("Failed to get available session.", e);
100. }
101. return retSession;
102. }

```

On lines 10 and 13, we see the `SessionParameters` argument being created, and the session ready wait timeout value being set to -1 which indicates that the session should be returned immediately. On line 17, the session is requested. On line 31, a `SessionStatusListenerProxy` is created. The `SessionStatusListenerProxy` requires an instance of a `SessionStatusListener`. This is supplied as an anonymous inner class starting on line 32.

Data Provider

Task: Retrieving a Data Provider Property Value

The Data Provider maintains a set of property values which reflect the current state of the POS application. The following are the steps involved in performing this task:

- Determine the category and property name.
- Retrieve the `POSDDataProvider` instance from the session.
- Use the `POSDDataProvider` to get the property value.

Determining the Category and Property Name

Use the tables in Appendix C. Data Provider Properties on page 280 to determine the property category and property name. For this example, we will retrieve the current transaction balance due. The category name will be the symbol `TransactionTotalsProperties.CATEGORY`, and the property name will be the symbol `TransactionTotalsProperties.AMOUNT_DUE`.

Retrieving the POSDataProvider Instance

Assuming that the variable "session" holds a valid session, the `POSDDataProvider` can be retrieved via the following code.

```

import com.ibm.retail.AEF.data.*;

POSDDataProvider dataProvider = null;
try
{
    dataProvider = session.getPOSDDataProvider();
}
catch (RemoteException re)
{
    // Some sort of network error happened.
}

```

Fetching the Property Value from the Data Provider

If the data provider instance has been retrieved successfully, then the property value may be retrieved with the following code:

```
String balanceDue = null;

try
{
    balanceDue = (String)(dataProvider.
        getPropertyValue(
            TransactionTotalsProperties.CATEGORY,
            TransactionTotalsProperties.AMOUNT_DUE));
}
catch (RemoteException re)
{
    // Some sort of network error happened.
}
```

SampleGUI Example

For a working example of fetching a POSDataProvider property value, see the `actionPerformed` method in the SampleGUI module `si/user/com/userco/retail/client/gui/CreditTenderAction.java`. In this method, the Data Provider property representing the transaction balance due is retrieved to be displayed in the dialog as shown:

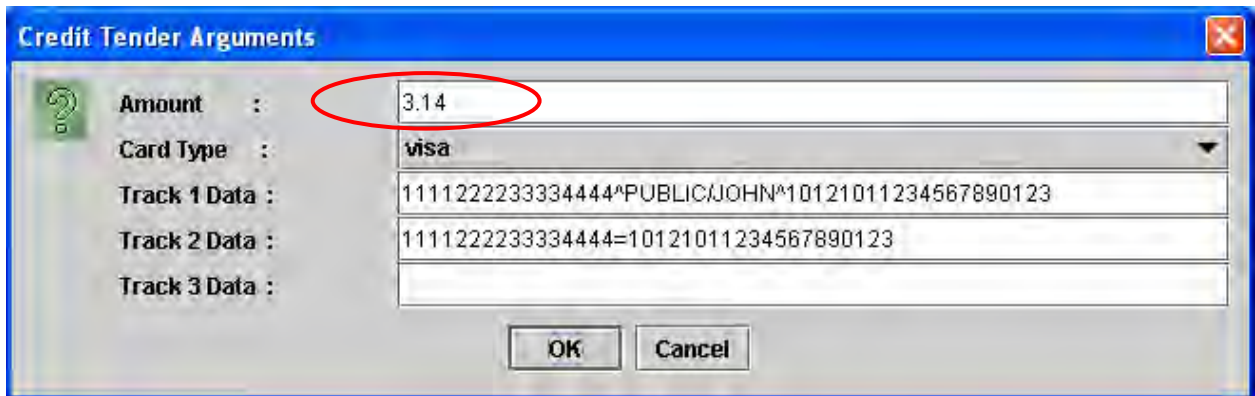


Figure 20 Displaying a Data Provider Property Value

Task: Subscribing to a Data Provider Event

The Data Provider publishes events triggered by the POS application. For a list of events, see

"Event and Listener Types" on page 47. The following are the steps for this task:

- Create a Java class which implements the desired listener interface.
- Use an instance of the listener to instantiate an instance of a proxy listener interface.

Creating the Listener Class.

For this example, we will create a ScaleListener. The ScaleListener can be implemented as a named, or unnamed class. This example will use a named ScaleListener called com.ibm.retail.AEF.sample.MyScaleListener. The following code for MyScaleListener must be copied to the com\ibm\retail\AEF\sample directory before compilation.

```
// com.ibm.retail.AEF.sample.MyScaleListener
import com.ibm.retail.AEF.data.*;
import com.ibm.retail.AEF.event.*;
import java.rmi.*;

public class MyScaleListener implements ScaleListener
{
    /**
     * Scale event occurred (an item was weighed).
     *
     * @param evt contains weight information
     * @throws RemoteException if remote access fails
     */
    public void itemWeighed(ScaleEvent evt) throws RemoteException
    {
        // User code to process the ScaleEvent goes here.
    }
}
```

Creating an Instance of the Proxy Listener

The proxy listener is created by passing the POSDataProvider for the session of interest, as well as an instance of the new listener class from the previous step.

```
import com.ibm.retail.AEF.client.*;
import com.ibm.retail.AEF.samples.*;

// Assume that dataProvider contains a valid reference to
// the session's data provider.

MyScaleListener myScaleListener = new MyScaleListener();

// Create the proxy (which also starts it listening).
try
{
    ScaleListenerProxy scaleProxy = new ScaleListenerProxy(
                                    dataProvider,
                                    myScaleListener);
}
catch (RemoteException re)
{
    // Network error.
}
catch (AEFException ae)
{
}
```

```

    // Failed to register listener with data provider.
}

// To stop the proxy from listening, use the following lines:
try
{
    scaleProxy.removeListener();
}
catch (RemoteException re)
{
    // Network error.
}
catch (AEFException ae)
{
    // Failed to remove listener from data provider.
}

```

SampleGUI Example

For a working example of subscribing to a POSDataProvider event, see the SampleGUI module `si/user/com/userco/retail/client/gui/StatusPanel.java`. This class subscribes to `AEFPropertyChangeEvent`s to display the current application state and substate as shown below.



Figure 21 AEFPropertyChangeListener Example

Task: Adding Additional Data to an Event

This task describes how to send additional information from the POS application to the AEF, and how to retrieve that information from a POSDataProvider event. For this task, we will assume that we have a customized version of SA or GSA which has added a time and attendance function to the operator logon and logoff operations. We will assume that if the user is clocking-in during a logon, then a global variable will be set to the clock-in time. In addition, another global variable will be set to the clock-out time if the operator clocks-out during the logoff.

The steps required to complete this task are:

- Write SA or GSA user exit code to add the additional data to the XML events being sent from the POS application to AEF.
- Write client code that listens for the POSDataProvider event and pulls out the new data.

Note for ACE users:

The process for modifying ACE is similar in principle to the process described above for SA and GSA. However, ACE does not make use of user exits as an extension model. Most ACE users have modified terminal sales programs, and for them, the normal make (build) process, using the ACE Toolkit, should be followed, including in the process the AEF interface module, DACEAEF.OBJ, to create a new terminal sales program. (DACEAEF.OBJ is included with Version 3 of ACE. ACE source code customers also receive DACEAEF.CPP, which is required if custom extensions are to be made to AEF or the Store Integrator GUI interface.)

Modifying the Events via User Exits

For this task, we wish to modify the XML events sent from the POS application to the AEF when an operator logs on or off. For format of these XML messages, see pages 316 and 318. For example, if an operator logon includes a clock-in, we want to add a "clockIn" attribute to the <signOn> message as shown below:

```
<signOn operatorID="887" operatorName="John Public"
clockIn="07/14/2004 08:35">
  <authorization ... />
</signOn>
```

In addition, if an operator logoff includes a clock-out, we want to add a "clockOut" attribute to the <signOff> message as shown below:

```
<signOff operatorID="887" clockOut="07/14/2004 14:58"/>
```

For SA and GSA, user exits have been provided which are called just before the XML message is sent to the AEF. These user exits allow modification of the XML message prior to sending the message to the AEF. We will show examples of the user exit code for SA and GSA below.

SA User Exit Code

SA includes the user exit module JAVAGUIU.BAS. This should be installed into c:/adx_upgm on the 4690 controller when you install the SA User Exit package. The relevant lines from this module are shown below:

```
%INCLUDE JGUIUVA.J86                                ! User variables

.
.
.

!-----
! subroutine javaAEFUserExit
!
! Params: eventNum - number of the event (application or user)
!          eventData - data associated with the event
```

```

!
!-----
SUB javaAEFUserExit(eventNum, eventData$) PUBLIC RECURSIVE !
javaGuiUserExit
    INTEGER*2 eventnum                ! event number
    STRING    eventData$              ! return data

    %INCLUDE JGUIU05.J86              ! allow user to
process

END SUB                                ! javaGuiUserExit

```

As shown above, the user exit subroutine is named `javaAEFUserExit`. This exit is called just before sending the XML message to the AEF. The `eventNum` indicates the cause of the event, and the `eventData$` variable contains the XML message itself.

The following table lists the values of the `eventNum` variable:

eventNum	Value
updateReceiptMsg	1
updateTotalMsg	2
updateReceiptMsg	3
updateScaleMsg	4
addAccountTenderMsg	5
addDiscountEntryMsg	6
addItemEntryMsg	7
addTenderEntryMsg	8
changeGivenMsg	9
endAccountingTransactionMsg	10
endSalesTransactionMsg	11
endTransactionMsg	12
operatorSignedOffMsg	13
operatorSignedOnMsg	14
optionsLoadedMsg	15
priceVerifyMsg	16
procedureCompletedMsg	17
startAccountingTransactionMsg	18
startSalesTransactionMsg	19
startTransactionMsg	20
reportOutputMsg	21
scrollMsg	22
displayVelocityCodesMsg	23
hideVelocityCodesMsg	24
redrawReceiptWindowMsg	25
updateStatusMsg	26
customerCardScannedMsg	27
javaKeysMsg	28
add2x20Msg	29
terminalSubStateMsg	30
optionsLoadingStartedMsg	31
optionsLoadingEndedMsg	32
optionsDataMsg	33
itemJustAddedMsg	34
voidTransactionMsg	35
foodStampTotalMsg	36
foreignTotalMsg	37

eventNum	Value
terminalConnectionChangeMsg	38
otrMsg	39
discountJustAddedMsg	40
operatorSpecialSignedOnMsg	41
updateTotalsDataMsg	42
aefEMredemptionCpnMsg	43
aefUnitOfWorkMsg	44
operatorSpecialSignedOffMsg	45
procedureStartedMsg2	46
operatorDisplay	47
customerDisplay	48
noSaleStartMsg	49
noSaleEndMsg	50
jGuiEndAppEvents	51

Table 11 SA User Exit Event Numbers

Note that only a subset of the events in the table cause SA to generate an XML message for the AEF. Therefore, not all the event numbers listed above will be passed through the `javaAEFUserExit` subroutine. In this case however, we will be checking for the `operatorSignedOnMsg` and `operatorSignedOffMsg` event numbers.

From the partial listing above, we see that the user variables must be declared in the module `JGUIUVA.J86`, and the logic for the `javaAEFUserExit` module must be placed in the module `JGUIU05.J86`. The complete code for these two modules is shown below.

```

!
!  jguiuva.j86
!

! insert user variable declarations here

STRING      GLOBAL
             CLOCKIN.TIME$,
             CLOCKOUT.TIME$
             \
             \ Set by user exit when operator
             \ logons on with clockin.
             ! Set by user exit when operator
             ! logs off with clockout.
```

```

!
! jguiu05.j86
!
!
! insert user code here

INTEGER*4 IDX,
      NDX

! Note that we are checking the eventNum, but we could also check
! the eventData$ itself to see if it is the XML message we are
! interested in modifying.

IF (eventNum = operatorSignedOnMsg) THEN BEGIN
  IF (LEN(CLOCKIN.TIME$)>0) THEN BEGIN
    ! We have a login with a clockin time, therefore we want
    ! to modify the XML string.

    ! Find the first ">" character.
    FOR IDX = 1 TO LEN(eventData$)
      IF MID$(eventData$,IDX,1) = ">" THEN BEGIN
        NDX = IDX
        IDX = LEN(eventData$)
      ENDIF
    NEXT IDX

    eventData$ = LEFT$(eventData$, NDX - 1) +
      " " +
      "clockIn=" +
      """" + CLOCKIN.TIME$ + """" +
      RIGHT$(eventData$, LEN(eventData$) - (NDX - 1))

  ENDIF
ENDIF

IF (eventNum = operatorSignedOffMsg) THEN BEGIN
  IF (LEN(CLOCKOUT.TIME$)>0) THEN BEGIN
    ! We have a login with a clockout time, therefore we want
    ! to modify the XML string.

    eventData$ = LEFT$(eventData$, LEN(eventData$) - 2) +
      " " +
      "clockOut=" +
      """" + CLOCKOUT.TIME$ + """" +
      RIGHT$(eventData$, 2)

  ENDIF
ENDIF

```

These modules are %INCLUDEd in JAVAGUIU.BAS, therefore JAVAGUIU.BAS must be recompiled. The resulting object module is embedded into JAVADEF1.L86, JAVADEF2.L86, and JAVADEF3.L86. Your terminal sales application includes one of these L86 modules. You can determine which module your terminal sales program uses by examining your EAMTS10L.INP module. You must regenerate the appropriate L86 module using the LIB86 command, and then relink your terminal sales module using the LINK86 command. More information on SA user exit programming may be found in the *4680-4690 Supermarket Application: Programmers Guide*.

GSA User Exit Code

The SI installation CD includes the user exit module EALAEFUE.BAS. This module should be copied into c:/adx_upgm on the 4690 controller. The relevant lines from this module are shown below:

```
!*****
function ?AEF.beforeXMLSend$(eventData$) public
    string    ?AEF.beforeXMLSend$
    string    eventData$

    ?AEF.beforeXMLSend$ = eventData$
end function
```

As shown above, the user exit subroutine is named ?AEF.beforeXMLSend\$. This exit is called just before sending the XML message to the AEF. The eventData\$ variable contains the XML message when may be modified by the user exit.

The following shows the completed code for the user exit module EALAEFUE.BAS:

```

! *****
!
! Module Name:          ealaefue.bas
!
!
! Descriptive Name:    User Exit CBASIC routines to handle AEF interface for GSA
!
!
! *****
! *****
%ENVIRON    T

STRING      GLOBAL                                \
            CLOCKIN.TIME$,                        \ Set by user exit when operator
                                                    \ logons on with clockin.
            CLOCKOUT.TIME$,                       ! Set by user exit when operator
                                                    ! logs off with clockout.

! *****
!
! *****
function ?AEF.beforeXMLSend$(eventData$) public
    string   ?AEF.beforeXMLSend$
    string   eventData$

INTEGER*4   IDX,                                \
            NDX

IF (LEFT$(eventData$,7) = "<signOn") THEN BEGIN
    IF (LEN(CLOCKIN.TIME$)>0) THEN BEGIN
        ! We have a login with a clockin time, therefore we want
        ! to modify the XML string.

        ! Find the first ">" character.
        FOR IDX = 1 TO LEN(eventData$)
            IF MID$(eventData$,IDX,1) = ">" THEN BEGIN
                NDX = IDX
                IDX = LEN(eventData$)
            ENDIF
        NEXT IDX

        eventData$ = LEFT$(eventData$, NDX - 1) +
            " " +
            "clockIn=" +
            """" + CLOCKIN.TIME$ + """" +
            RIGHT$(eventData$, LEN(eventData$) - (NDX - 1))

    ENDIF
ENDIF

IF (LEFT$(eventData$,8) = "<signOff") THEN BEGIN
    IF (LEN(CLOCKOUT.TIME$)>0) THEN BEGIN
        ! We have a login with a clockout time, therefore we want
        ! to modify the XML string.

        eventData$ = LEFT$(eventData$, LEN(eventData$) - 2) +
            " " +
            "clockOut=" +
            """" + CLOCKOUT.TIME$ + """" +
            RIGHT$(eventData$, 2)

    ENDIF
ENDIF

?AEF.beforeXMLSend$ = eventData$
end function

```

This module must be recompiled. If you followed the installation and configuration instructions for AEF related to GSA, you have already modified the terminal sales .INP file to include the object module resulting from compiling EALAEFUE.BAS. Therefore, you need only relink your terminal sales application to include the module. More information on GSA user exit programming may be found in the *4680-4690 General Sales Application: Programmers Guide*.

Retrieving the New Event Data

Now that we have modified the POS application to include additional fields in the XML message which is sent to the AEF, we show how to retrieve the data out of the POSDataProvider event. No changes are necessary to the POSDataProvider, or to the relevant event classes. The additional data is automatically included in the relevant POSDataProvider events.

For this example, the “clockIn” and “clockOut” attributes will be included in the POSDataProvider OperatorEvent. We can retrieve these attribute values from the OperatorEvent by using the POSAppEventElement.getProperty method as explained in “Extending POSAppEvents” on page 52.

When we code our OperatorListener.operatorEventOccurred method, we can retrieve the “clockIn” and “clockOut” attributes (if they exist) as follows:

```
public void operatorEventOccurred(OperatorEvent evt)
{
    String clockIn = null;
    String clockOut = null;

    clockIn = evt.getProperty("clockIn");
    if (clockIn != null)
    {
        // Process the clockIn here.
    }

    clockOut = evt.getProperty("clockOut");
    if (clockOut != null)
    {
        // Process the clockOut here.
    }
}
```

Task: Sending a New XML Event

This task describes how to send a completely new XML message from the POS application to the AEF. This may be necessary for AEF enabling a user extension or a feature not already AEF enabled. For this task, we will assume that the POS application needs to be extended to keep the AEF updated with the current tender positions of the till.

Note: We assume that the user is familiar with the extension mechanism relevant to the POS application.

The steps required to complete this task are:

- Define the XML message format.
- Modify the POS application to send the new message when appropriate.
- Implement a new Event, Listener, ListenerSupport, and ListenerProxy classes for the new event.
- Create a class which implements the SessionExtension interface. This class will be responsible for registering the new ListenerSupport class with the POSDataProvider.
- Configure the Data Provider to create an instance of the new event class whenever it sees the new XML message from the POS application.
- Register to receive the event in the client. This involves using the new ListenerProxy class.
- Package and distribute the new modules.

Defining the XML message

For this task, assume that the XML message will be of the form

```
<tillPosition cash="nnnnn.nn" checks=" nnnnn.nn" foodstamps="
nnnnn.nn"/>
```

Modify the POS Application to Send the Message

Use the standard extension technique for the POS application (such as user exit programming for SA and GSA) to send the XML message when appropriate. The XML message must be constructed as a string.

GSA Code for Sending the XML Message

To send the XML message from GSA, include the following function declaration

```
FUNCTION ?AEF.EVENT(eventData$) EXTERNAL          ! From ealeafc
    STRING          eventData$                     ! Event Data
END FUNCTION
```

The XML message should be placed in eventData\$, and the ?AEF.EVENT function should be called to send the event to the AEF.

SA Code for Sending the XML Message

The following code should be used to define a function called `send.XML.Message` which can be used to send the XML message to the AEF.

```
%INCLUDE JAVAGUIV.J86                                ! Global Variable include

FUNCTION javaHealth EXTERNAL
END FUNCTION

FUNCTION aefBlockForJava EXTERNAL
END FUNCTION
```

```

SUB transmitToJVM(eventNum, temp$, eventData$, eventSource) EXTERNAL
  INTEGER*2
    eventNum,
    eventSource

  STRING
    eventData$,
    temp$
END SUB

FUNCTION send.XML.Message(XML.MESSAGE$) PUBLIC RECURSIVE

  String
    XML.MESSAGE$

  INTEGER*2
    sendEvent

  sendEvent = 0

  IF (JAVA.INIT = -1) THEN BEGIN
    IF (javaHealth = -1) THEN BEGIN
      sendEvent = -1
    ENDIF

    ! If command tail option specifies that we should block
    ! then the next line will block until a pipe created in
    ! the JVM is detected. If the blockForJava options is not
    ! set, or the stub version of the aef hooks is linked in,
    ! the call will return a zero.
    sendEvent = aefBlockForJava
    IF (sendEvent = 0) THEN BEGIN
      IF (javaHealth = -1) THEN BEGIN
        sendEvent = -1
      ENDIF
    ENDIF
    IF (sendEvent = -1) THEN BEGIN
      IF (callAEFHooks) THEN BEGIN
        javaGuiClass$ =
          "com.ibm.retail.AEF.data.ApplicationDataConnectorImpl" !
set the class
        javaGuiMethod$ = "processEvent" ! set the method
        CALL transmitToJVM(0, XML.MESSAGE$, XML.MESSAGE$, 2)
      ENDIF
    ENDIF
  ENDIF

END FUNCTION

```

Implement the New Event, Listener, and ListenerSupport Classes

We will define the following interfaces and classes:

- TillPositionEvent
- TillPositionEventImpl
- TillPositionListener
- TillPositionListenerSupport
- TillPositionListenerProxy

TillPositionEvent

The module

d:/si/user/com/userco/retail/AEF/event/TillPositionEvent.java can be written as follows:

```
/*
 * TillPositionEvent Interface
 *
 * 07/07/04
 *
 * Copyright:
 * Licensed Materials - Property of IBM
 * "Restricted Materials of IBM"
 * 5724-AEF
 * (C) Copyright IBM Corp. 2004.
 */
package com.userco.retail.AEF.event;

import com.ibm.retail.AEF.event.*;

/**
 * This class is a sample user event.
 */

public interface TillPositionEvent extends POSAppEvent
{
    // Note, we want these constants to be the same as the attribute
    // names in the XML
    public static final String CASH_POSITION="cash";
    public static final String CHECK_POSITION="checks";
    public static final String FOODSTAMP_POSITION="foodstamps";

    /**
     * Get the cash position.
     *
     * @return String
     */
    public String getCashPosition();

    /**
     * Get the foodstamps position.
     *
     * @return String
     */
    public String getFoodStampPosition();

    /**
     * Get the check position.
     *
     * @return String
     */
    public String getCheckPosition();
}
}
```

TillPositionEventImpl

The module

d:/si/user/com/userco/retail/AEF/event/TillPositionEventImpl.java can be written as follows:

```
/*
 * TillPositionEventImpl
 *
```

```

* 07/07/04
*
* Copyright:
* Licensed Materials - Property of IBM
* "Restricted Materials of IBM"
* 5724-AEF
* (C) Copyright IBM Corp. 2004.
*
*/
package com.userco.retail.AEF.event;

import com.ibm.retail.si.util.*;
import com.ibm.retail.AEF.data.*;
import com.ibm.retail.AEF.event.*;

import org.xml.sax.*;

/**
 * An example of a user event.
 */

public class TillPositionEventImpl extends POSAppEventImpl implements
TillPositionEvent
{
    static String copyright()
    { return com.ibm.retail.si.Copyright.IBM_COPYRIGHT_SHORT; }

    /**
     * Construct an TillPositionEventImpl
     */
    public TillPositionEventImpl()
    {
        super();
    }

    /**
     * Sets default property values.
     */
    public void setDefaultProperties()
    {
        //properties.put("????", new Boolean(false));
    }

    /**
     * Get the cash position.
     * @return String
     */
    public String getCashPosition()
    {
        return (String) (properties.get(TillPositionEvent.CASH_POSITION));
    }

    /**
     * Get the foodstamps position.
     * @return String
     */
    public String getFoodStampPosition()
    {
        return
        (String) (properties.get(TillPositionEvent.FOODSTAMP_POSITION));
    }
}

/**

```

```

        * Get the check position.
        *
        * @return String
        */
public String getCheckPosition()
{
    return (String) (properties.get(TillPositionEvent.CHECK_POSITION));
}

/**
 * Gets the AEF property change category defined for this event.
 * @return String category id
 */
public String getPropertyChangeCategory()
{
    return WorkstationStatusProperties.CATEGORY;
}

// Instance Variables
}

```

TillPositionListener

The module

d:/si/user/com/userco/retail/AEF/event/TillPositionListener.java
can be written as follows:

```

/*
 * TillPositionListener Interface
 *
 * 07/07/04
 *
 * Copyright:
 * Licensed Materials - Property of IBM
 * "Restricted Materials of IBM"
 * 5724-AEF
 * (C) Copyright IBM Corp. 2004.
 */
package com.userco.retail.AEF.event;

import com.ibm.retail.AEF.event.*;
import java.rmi.*;

/**
 * Listener interface for receiving <code>TillPositionEvents</code> from the
 * terminal session.
 */

public interface TillPositionListener extends POSAppEventListener
{
    /**
     * The till contents have changed.
     *
     * @param evt
     */
    public void tillPositionChanged(TillPositionEvent evt) throws
    RemoteException;
}

```

TillPositionListenerSupport

The module

d:/si/user/com/userco/retail/AEF/event/TillPositionListenerSupport.java helps the POSDataProvider determine the listener type and listener method to call for a specific event type. The module can be written as follows:

```
/*
 * TillPositionListenerSupport Interface
 *
 * 07/07/2004
 *
 * Copyright:
 * Licensed Materials - Property of IBM
 * "Restricted Materials of IBM"
 * 5724-AEF
 * xxxx-xxx
 * (C) Copyright IBM Corp. 2004.
 */
package com.userco.retail.AEF.event;

import java.util.*;
import java.rmi.*;

import com.ibm.retail.AEF.util.*;
import com.ibm.retail.AEF.event.*;
import com.ibm.retail.si.util.*;
import com.ibm.retail.si.util.AEFConst;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

/**
 * ListenerSupport objects are used internally by the POSDataProvider
 * to handle notification of events received.
 */
public class TillPositionListenerSupport extends POSAppEventListenerSupport
{
    private static Log log =
LogFactory.getLog(TillPositionListenerSupport.class);

    /**
     * Add a listener to be notified of events handled by this support class
     *
     * @param listener
     * @throws AEFEException if listener is not of proper type
     */
    public void addPOSAppEventListener(POSAppEventListener listener) throws
AEFEException
    {
        if (listener instanceof TillPositionListener)
        {
            super.addPOSAppEventListener(listener);
            if (log.isTraceEnabled())
            {
                log.trace("Added listener");
            }
        }
        else
        {
            throw (new AEFEException(AEFConst.INVALID_LISTENER_TYPE,
listener.getClass().getName()));
        }
    }

    /**
     * Notify an event listener
     */
}
```

```

*
* @param listener
* @param event
* @throws AEFEException if notification fails
*
*/
public void notifyListener(POSAppEventListener listener, POSAppEvent evt)
throws AEFEException
{
    if (evt instanceof TillPositionEvent)
    {
        TillPositionListener l = (TillPositionListener)listener;
        try
        {
            if (log.isTraceEnabled())
            {
                log.trace("Notify listener: " + evt.toString() );
            }
            l.tillPositionChanged((TillPositionEvent)evt);
        }
        catch (Exception e)
        {
            handleListenerException(listener, e);
        }
    }
    else
    {
        // improper event type - throw an exception
        throw (new AEFEException(AEFEConst.EVENT_LISTENER_MISMATCH,
            evt.getClass().getName()));
    }
}

/**
* Returns class name of listeners that are supported
*
* @return listener class name
*
*/
public String getListenerClassName()
{
    return (TillPositionListener.class.getName());
}

/**
* Returns class name of POSAppEvents that are supported
*
* @return event class name
*
*/
public String getEventClassName()
{
    return (TillPositionEventImpl.class.getName());
}
}

```

The four modules above can be compiled with the command

```

d:/si/user>javac -classpath
d:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:/si/si_jars/siutil
.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar
com/userco/retail/AEF/event/*.java

```

TillPositionListenerProxy

The module

d:/si/user/com/userco/retail/AEF/client/TillPositionListenerProxy.
java is an RMI object which helps localize event dispatching in a remote client. This module requires a special compile step since it is an RMI object. The module can be written as:

```
/*
 * TillPositionListenerProxy
 *
 * 07/074/04
 *
 * Copyright:
 * Licensed Materials - Property of IBM
 * "Restricted Materials of IBM"
 * 5724-AEF
 * (C) Copyright IBM Corp. 2003.
 */
package com.userco.retail.AEF.client;

import com.ibm.retail.AEF.event.*;
import com.ibm.retail.AEF.data.*;
import com.ibm.retail.AEF.client.*;
import com.ibm.retail.AEF.util.*;
import com.ibm.retail.si.util.*;
import com.ibm.retail.si.Copyright;

import com.userco.retail.AEF.event.*;

import java.rmi.*;
import java.rmi.server.*;
import java.lang.reflect.*;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

/**
 * The TillPositionListenerProxy provides a proxy object for remote listeners
 * to monitor the TillPosition events of its associated terminal session.
 * <p>
 * Remote proxy objects extend the appropriate RMI server classes which handle
 the
 * RMI duties for the client (so the client does not need to deal with rmi
 server issues).
 * The proxy objects also perform the role of event dispatching.
 * <p>
 * By default, the listener proxy object will perform event queuing on an AEF
 event thread.
 * This relieves the client of any "thread swapping" responsibilities and
 insures that the
 * AEF event dispatching performance is not affected by client processing. To
 override this
 * default behavior, use the <i>setDispatchQueue()</i> method. The proxy
 utilizes the
 * EventDispatcher for queueing event listener notification.
 * <p>
 * To use a proxy object, the client must implement the listener interface
 * and get the POSDataProvider object from the AEFSession to monitor. The proxy
 * object is then constructed passing a reference to the client and the data
 provider.
 * The proxy object performs the listener registration and forwards
 * all events from the data provider to the client.
 */

public class TillPositionListenerProxy extends AEFEventListenerProxy implements
AEFListenerProxyInterface,

TillPositionListener
{
```

```

static String copyright()
{ return com.ibm.retail.si.Copyright.IBM_COPYRIGHT_SHORT; }

private static Log log = LogFactory.getLog(TillPositionListenerProxy.class);

protected static Method listenerMethod =
getListenerMethod(TillPositionEvent.class,TillPositionListener.class,"tillPositi
onChanged");

/**
 * Construct an TillPositionListenerProxy.
 * This method registers the proxy as a listener with the POSDataProvider and
begins event
 * notification to the proxy client listener.
 *
 * @param dataProvider instance of POSDataProvider which is the source of
events
 * @param listener client listener which receives the event notification via
this proxy
 *
 * @exception java.rmi.RemoteException
 * @exception AEFException
 * Among the possible AEFException error codes are:
 * <br>AEFConst.NO_LISTENER_SUPPORT
 */
public TillPositionListenerProxy(POSDataProvider dataProvider,
TillPositionListener listener) throws RemoteException, AEFException
{
    this.dataProvider = dataProvider;
    this.listener = listener;

dataProvider.addPOSAppEventListener(TillPositionListener.class.getName(),this);
}

/**
 * Remove the proxy listener from the POSDataProvider. Ends notification of
events.
 *
 * @exception java.rmi.RemoteException
 * @exception AEFException
 * Among the possible AEFException error codes are:
 * <br>AEFConst.NO_LISTENER_SUPPORT
 */
public void removeListener() throws RemoteException, AEFException
{
dataProvider.removePOSAppEventListener(TillPositionListener.class.getName(),this
);
}

/**
 * Till position changed on the POS terminal session.
 * Dispatches event notification to the proxy client listener.
 *
 * @param evt
 * @throws RemoteException if remote access fails
 */
public void tillPositionChanged(TillPositionEvent evt) throws RemoteException
{
    dispatchEvent(evt,listener,listenerMethod);
}

// instance data
protected POSDataProvider dataProvider;
protected TillPositionListener listener;
}

```

The `TillPositionListenerProxy` must be compiled with the following **two** commands:

```
d:\si\user>javac -classpath
.;d:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:/s
i/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar
com\userco\retail\AEF\client\*.java
```

```
d:\si\user>rmic -classpath
.;c:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:/si
/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar
com.userco.retail.AEF.client.TillPositionListenerProxy
```

Create a SessionExtension to Register the ListenerSupport Object

An instance of the new `TillPositionListenerSupport` class must be registered with the `POSDDataProvider`. We can make use of the [SessionExtension](#) to accomplish this. Classes which implement the `SessionExtension` interface may be configured to be created when the session is created. An instance of the class is created, and passed a reference to the session. We will use this feature to register an instance of our new `TillPositionListenerSupport` class.

We will create the class file

`d:/si/user/com/userco/retail/AEF/session/UserListenerSupportRegistrar.java` with the following contents:

```
/*
 * UserListenerSupportRegistrar
 *
 * 07/07/04
 *
 * Copyright:
 * Licensed Materials - Property of IBM
 * "Restricted Materials of IBM"
 * 5724-AEF
 * xxxx-xxx
 * (C) Copyright IBM Corp. 2004.
 */
package com.userco.retail.AEF.session;

import com.userco.retail.AEF.event.*;
import com.ibm.retail.AEF.data.*;
import com.ibm.retail.AEF.session.*;

import java.rmi.*;

/**
 * This class is configured to be created when the session is created.
 * We use it to register our TillPositionListenerSupport instance with
 * the
 * POSDataProvider.
 */

public class UserListenerSupportRegistrar implements SessionExtension
{
    /**
     * Constructor
     */
    public UserListenerSupportRegistrar()
    {
        tillPositionListenerSupport = new TillPositionListenerSupport();
    }
}
```

```

    }

    /**
     * This method will be called after the session becomes ready.
     *
     * @param session The AEFSession associated with this extension
     * object.
     */
    public void setSession(AEFSession session)
    {
        POSDataProvider dataProvider = null;
        try
        {
            dataProvider=session.getPOSDataProvider();

dataProvider.addPOSAppEventListenerSupport(tillPositionListenerSupport);
        }
        catch (RemoteException re)
        {
            // Impossible to get here, because the call is local.
        }
    }

    // Instance variables
    protected TillPositionListenerSupport tillPositionListenerSupport;
}

```

This module may be compiled with the command:

```

d:\si\user>javac -classpath
.;d:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:/si/si_jars/siut
il.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar
com\userco\retail\AEF\session\*.
java

```

We must configure the AEF to create an instance of our new UserListenerSupportRegistrar class by creating or editing the file d:/si/userjar/usersession.properties and adding the line:

```

session.extensions=com.userco.retail.AEF.session.UserListenerSuppo
rtRegistrar

```

Configuring Data Provider to Create the New Event

In order to configure the Data Provider to create and fire an instance of the new TillPositionEventImpl class, we must add the XML element tag "tillPosition" as a key in the classes.properties bundle. To do this, create or edit the file d:/si/userjar/userclass.properties and add the line:

```

tillPosition=com.userco.retail.AEF.event.TillPositionEventImpl

```

Registering the Client for the New Event

The client should use the new TillPositionListenerProxy in order to receive and process the new TillPositionEvents. The following is a code fragment that shows how a client would create a TillPositionListenerProxy:

```

// Attach a TillPositionListener to process the TillPositionEvents.
// Assume that "session" holds a valid AEFSession.
TillPositionListenerProxy listenerProxy = null;
try
{
    listenerProxy = new
        TillPositionListenerProxy(session,
                                new TillPositionListener()
        {
            /**
             * Indicates till position has changed.
             *
             * @param evt contains details of TillPositionEvent
             */
            public void tillPositionChanged(TillPositionEvent evt)
            {
                // Process the new till position here!
                System.out.println("New cash position = " +
                                evt.getCashPosition());
            }
        }
    );
}
catch (Throwable th)
{
    log.error("Unable to add TillPositionListener.", th);
}

```

Packaging the New Modules

The user class and properties files need to be bundled into the appropriate user*.jar file, copied into the 4690 c:\adx_ipgm directory, and distributed. This may require that CSS and the registers be stopped. The jar file also needs to be included on the classpath in the client environment.

To create a new user.jar file, issue the command

```
d:/si/userjar>jar cfv user.jar *
```

Automation Provider

Task: Using an AEFSession

This task illustrates the use of an AEFSession. After acquiring the session, the example shows logging onto the session, starting a transaction, adding an item, suspending the transaction, and logging off the session.

The steps involved in this task are

- Write the `AEFExample3` class, which will get an AEFSession, and use the POSAutomation API to initialize the session, logon to the session, start a transaction, add an item to the transaction, suspend the transaction, then logoff the session.
- Compile the class.
- Execute the client code, assuming that CSS is running with virtual sessions on a 4690 within the same subnet as the client.

Note that this example uses default operator id and passwords to logon. Therefore these must be configured as described in “Default Operator ID/Passwords” on page 140.

Writing the AEFExample3 Class

A complete `AEFExample3` class has been provided in `d:\si\user\com\userco\retail\client\example3\AEFExample3.java`. Examine the source code to see how the AEFSession is used.

Compiling the Code

The code has already been compiled, but if you want to modify it, you can recompile it with the command:

```
d:\si\user>javac -classpath
d:/si/si_jars/aef.jar;d:/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar
com/userco/retail/client/example3/AEFExample3.java
```

Executing the Code

To execute the code, execute the command:

```
d:\si\user> java -Djava.rmi.server.hostname=xx.xx.xx.xx -classpath .;d:/si/si_j
ars/aef.jar;d:/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar;d:/si_jars/simgm
t.jar com.userco.retail.client.example1.AEFExample3
```

Where `xx.xx.xx.xx` is the TCIP/IP address of the client network interface card which is on the same network as the `SessionServer`.

When the program is executed, its output will be displayed on the console window.

Using an AEFSession – SampleGUI

The main screen of the SampleGUI application contains a button palette. Each button is associated with an action, and many of these actions use POSAutomationProvider API methods to interact with the AEFSession.

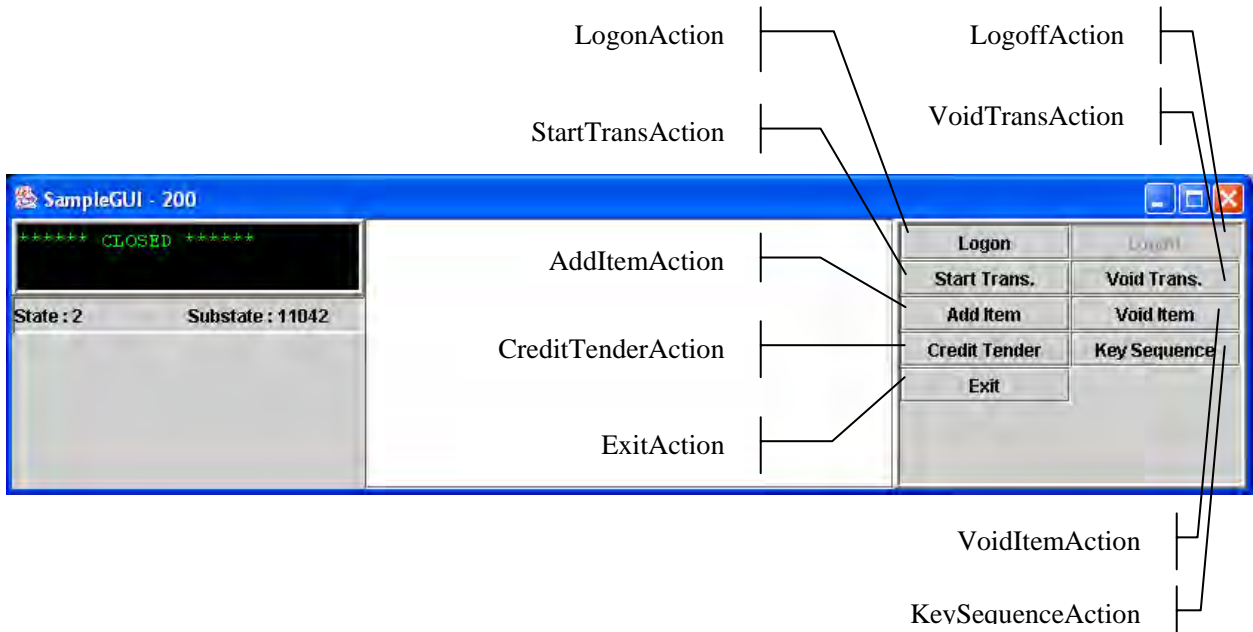


Figure 22 SampleGUI Action Classes

Each of the action classes in the figure above call a method in `d:/si/user/com/userco/retail/client/gui/Controller.java`. This class demonstrates using POSAutomationProvider methods to interact with an AEFSession. For example, the `LogonAction` calls the `Controller.logon` method as shown on line # 111 below.

```
1. /*
2.  * Controller
3.  *
4.  * 07/07/04
5.  *
6.  * Copyright:
7.  * Licensed Materials - Property of IBM
8.  * "Restricted Materials of IBM"
9.  * 5724-AEF
10. * (C) Copyright IBM Corp. 2004.
11. *
12. */
13. package com.userco.retail.client.gui;
14.
15. import com.ibm.retail.AEF.session.*;
16. import com.ibm.retail.AEF.automation.*;
17. import com.ibm.retail.AEF.data.*;
18. import com.ibm.retail.AEF.workstation.*;
19. import java.util.logging.*;
20. import java.util.*;
21. import java.lang.ref.*;
22.
23. import com.ibm.retail.si.Copyright;
24.
25. import org.apache.commons.logging.Log;
26. import org.apache.commons.logging.LogFactory;
```

```

27.
28. /**
29.  * This class is used as the controller interface to the AEF Automation API.
30.  * All SampleGUI calls to the automation API go through the controller.
31.  */
32. public class Controller
33. {
34.
35.     static String copyright()
36.     { return com.ibm.retail.si.Copyright.IBM_COPYRIGHT_SHORT; }
37.
38.     private static Log log = LogFactory.getLog(Controller.class);
39.
40.
41.     /**
42.     * Default constructor.
43.     *
44.     * @param boolean True if the requested session was a specific
45.     *                terminal number. False if the session was
46.     *                any available session.
47.     */
48.     public Controller(boolean specificSession)
49.     {
50.         specific = specificSession;
51.     }
52.
53.     /**
54.     * Sets the session.
55.     *
56.     * @param session
57.     */
58.     public void setSession(AEFSession session)
59.     {
60.         this.session = session;
61.         try
62.         {
63.             automation = session.getPOSAutomationProvider();
64.         }
65.         catch (Throwable th)
66.         {
67.             log.error("Unable to retrieve POSAutomationProvider.",th);
68.         }
69.     }
70.
71.     /**
72.     * Gets the session.
73.     *
74.     * @return AEFSession
75.     */
76.     public AEFSession getSession()
77.     {
78.         return session;
79.     }
80.
81.     /**
82.     * Gets the transaction if one is in progress, otherwise returns null.
83.     *
84.     * @return Transaction
85.     */
86.     public Transaction getTransaction()
87.     {
88.         Transaction retVal = null;
89.         if (automation == null)
90.         {
91.             log.error("Cannot get transaction because POSAutomationProvider API is
92.             unavailable.");
93.         }
94.         else
95.         {
96.             try
97.             {
98.                 retVal = automation.getTransaction();

```

```

98.         }
99.         catch (Throwable th)
100.        {
101.            log.error("Unable to retrieve transaction from POSAutomationProvider.",
102.            th);
103.        }
104.        return retVal;
105.    }

106.    /**
107.     * Logs onto the session with the OperatorIdentifier arguments.
108.     *
109.     * @return Operator
110.     */
111.    public Operator logon(OperatorIdentifier opIdent)
112.    {
113.        Operator retVal = null;
114.
115.        if (automation == null)
116.        {
117.            log.error("Cannot logon because POSAutomationProvider API not
118.            available.");
119.        }
120.        else
121.        {
122.            try
123.            {
124.                retVal = automation.logon(opIdent);
125.            }
126.            catch (Throwable th)
127.            {
128.                // Perform error handling here.
129.                log.error("Unable to logon.", th);
130.            }
131.        }
132.        return retVal;
133.    }
134.    /**
135.     * Logs off the session.
136.     *
137.     */
138.    public void logoff()
139.    {
140.        if (automation == null)
141.        {
142.            log.error("Cannot logoff because POSAutomationProvider API not
143.            available.");
144.        }
145.        else
146.        {
147.            try
148.            {
149.                automation.logoff();
150.            }
151.            catch (Throwable th)
152.            {
153.                // Perform error handling here.
154.                log.error("Unable to logoff.", th);
155.            }
156.        }
157.    }
158.    /**
159.     * Starts a regular sales transaction.
160.     *
161.     * @return SalesTransaction
162.     */
163.    public SalesTransaction startTransaction(TransactionIdentifier transIdent)
164.    {

```

```

165.     SalesTransaction retVal = null;
166.     if (automation == null)
167.     {
168.         log.error("Cannot start transaction because POSAutomationProvider API not
available.");
169.     }
170.     else
171.     {
172.         try
173.         {
174.             retVal = (SalesTransaction)(automation.startTransaction(transIdent));
175.         }
176.         catch (Throwable th)
177.         {
178.             // Perform error handling here.
179.             log.error("Unable to start transaction.", th);
180.         }
181.     }
182.     return retVal;
183. }
184.
185. /**
186.  * Voids the transaction in progress.
187.  *
188.  */
189. public void voidTransaction()
190. {
191.     if (automation == null)
192.     {
193.         log.error("Cannot void transaction because POSAutomationProvider API not
available.");
194.     }
195.     else
196.     {
197.         try
198.         {
199.             automation.voidCurrentTransaction();
200.         }
201.         catch (Throwable th)
202.         {
203.             // Perform error handling here.
204.             log.error("Unable to void transaction.", th);
205.         }
206.     }
207. }
208.
209. /**
210.  * Releases the session.
211.  *
212.  */
213. public void releaseSession()
214. {
215.     if (!specific)
216.     {
217.         // We only need to release the session if it was gotten
218.         // via a call to getAvailableSession.
219.         try
220.         {
221.             session.release();
222.         }
223.         catch (Throwable th)
224.         {
225.             // Perform error handling here.
226.             log.error("Unable to release the session.", th);
227.         }
228.     }
229. }
230.
231. /**
232.  * Adds one or more items to the sales transaction.
233.  *
234.  * @param itemIdent The ItemIdentifier containing arguments such

```

```

235.     * as the itemcode, quantity, etc.
236.     * @return ArrayList of LineItems.
237.     */
238.     public ArrayList addItem(ItemIdentifier itemIdent)
239.     {
240.         ArrayList retVal = null;
241.         if (automation == null)
242.         {
243.             log.error("Cannot add items because POSAutomationProvider API not
available.");
244.         }
245.         else
246.         {
247.             SalesTransaction trans = (SalesTransaction) (getTransaction());
248.             if (trans == null)
249.             {
250.                 log.error("Cannot add items because a transaction is not in
progress.");
251.             }
252.             else
253.             {
254.                 try
255.                 {
256.                     retVal = trans.addItem(itemIdent);
257.                 }
258.                 catch (Throwable th)
259.                 {
260.                     // Perform error handling here.
261.                     log.error("Unable to add item.", th);
262.                 }
263.             }
264.         }
265.         return retVal;
266.     }
267.
268. /**
269.  * Voids an item from the sales transaction.
270.  *
271.  * @param itemIdent The ItemIdentifier containing arguments such
272.  * as the itemcode, quantity, etc.
273.  */
274.     public void voidItem(ItemIdentifier itemIdent)
275.     {
276.         if (automation == null)
277.         {
278.             log.error("Cannot add items because POSAutomationProvider API not
available.");
279.         }
280.         else
281.         {
282.             SalesTransaction trans = (SalesTransaction) (getTransaction());
283.             if (trans == null)
284.             {
285.                 log.error("Cannot void item because a transaction is not in
progress.");
286.             }
287.             else
288.             {
289.                 try
290.                 {
291.                     trans.voidItem(itemIdent);
292.                 }
293.                 catch (Throwable th)
294.                 {
295.                     // Perform error handling here.
296.                     log.error("Unable to void item.", th);
297.                 }
298.             }
299.         }
300.     }
301.
302. /**

```

```

303.    * Adds a tender to the sales transaction.
304.    *
305.    * @param creditIdent The MSRCreditIdentifier
306.    * containing arguments such
307.    * as the tender amount, card type and track data.
308.    * @return ArrayList of LineItems.
309.    */
310.    public ArrayList addTender(MSRCreditIdentifier creditIdent)
311.    {
312.        ArrayList retVal = null;
313.        if (automation == null)
314.        {
315.            log.error("Cannot add tender because POSAutomationProvider API not
available.");
316.        }
317.        else
318.        {
319.            SalesTransaction trans = (SalesTransaction) (getTransaction());
320.            if (trans == null)
321.            {
322.                log.error("Cannot add items because a transaction is not in
progress.");
323.            }
324.            else
325.            {
326.                try
327.                {
328.                    retVal = trans.addTender(creditIdent);
329.                }
330.                catch (Throwable th)
331.                {
332.                    // Perform error handling here.
333.                    log.error("Unable to add tender.", th);
334.                }
335.            }
336.        }
337.        return retVal;
338.    }
339.
340.    /**
341.    * Sends a key sequence to the POS application.
342.    *
343.    * @param sequence The key sequence to be sent. Function codes must
344.    * be inside < > characters.
345.    */
346.    public void keySequence(String sequence)
347.    {
348.        Workstation workstation = null;
349.        try
350.        {
351.            workstation = session.getWorkstation();
352.        }
353.        catch (Throwable th)
354.        {
355.            log.error("Unable to retrieve Workstation to send the key sequence.",
th);
356.            return;
357.        }
358.        try
359.        {
360.            workstation.sendKeySequence(sequence);
361.        }
362.        catch (Throwable th2)
363.        {
364.            log.error("Unable to send key sequence to POS application.", th2);
365.        }
366.    }
367.
368.    // Member variables
369.    protected AEFSession session;
370.    protected POSAutomationProvider automation;
371.    protected boolean specific;

```

Task: Extending Existing Automation Provider APIs

Because of user enhancements or modifications to the POS application, it may be necessary to extend existing POSAutomationProvider API's to handle additional arguments. For this task, we will assume that the POS application is SA, and that the start transaction key sequence has been modified from "0 <ENTER>" to "0 <SLASH> *zipcode* <ENTER>".

This type of key sequence modification implies a modification to the SA state table, as well as to the SA application to deal with the additional data. However, discussion of state table and SA user exit programming is beyond the scope of this example, and it is assumed that these steps have already been performed.

The remaining steps necessary for this task are:

- Add the additional zipcode argument to the TransactionIdentifier used to start the transaction.
- Replace the implementation of SASStartTransactionActionImpl.java to use the additional data.
- Modify the key sequence to override the start transaction key sequence definition.
- Bundle the new files in a user.jar file, install and distribute on the 4690.

Adding Additional API Arguments

Assume that the client starts a transaction with the following code:

```
TransactionIdentifier transIdentifier = new
    TransactionIdentifierImpl();
transIdent.setTransactionType(AEFConst.REGULAR_SALE);
SalesTransaction tran = null;
try
{
    tran = automation.startTransaction(transIdentifier);
}
catch (RemoteException re)
{
    // Handle the network problem here.
}
catch (AEFException ae)
{
    // Handle the AEF problem here.
}
```

We can take advantage of the fact that the TransactionIdentifier is a HashMap to add the additional argument as follows:

```
String zipcode = "27607";
```

```
TransactionIdentifier transIdentifier = new
    TransactionIdentifierImpl();
transIdent.setTransactionType(AEFConst.REGULAR_SALE);
transIdent.put("zipcode", zipcode);
SalesTransaction tran = null;
try
{
    tran = automation.startTransaction(transIdentifier);
}
catch (RemoteException re)
{
    // Handle the network problem here.
}
catch (AEFException ae)
{
    // Handle the AEF problem here.
}
```

Implementing the Action

Next, we look at the Javadoc for the [startTransaction](#) method to see that the Action ID for the method is "StartTransaction". By looking at "StartTransaction" on page 170, we see the implementing action is [SAStartTransactionActionImpl](#). We need to override this class to do two things:

1. In the constructor, we want to retrieve and save the zipcode from the TransactionIdentifier.
2. In the `sendStartTransactionSequence` method, we need to add the zipcode to the key sequence arguments.

To achieve these two steps, we create a new class called `d:/si/user/com/userco/retail/AEF/action/UserStartTransactionActionImpl.java`. This class extends `SAStartTransactionActionImpl.java` and is shown below.

```
1.  /*
2.   * UserStartTransactionActionImpl
3.   *
4.   * 07/07/2004
5.   *
6.   * Copyright:
7.   * Licensed Materials - Property of IBM
8.   * "Restricted Materials of IBM"
9.   * 5724-AEF
10.  * (C) Copyright IBM Corp. 2004.
11.  *
12.  */
13. package com.userco.AEF.action;
14.
15. import com.ibm.retail.AEF.automation.*;
16. import com.ibm.retail.AEF.util.*;
17. import com.ibm.retail.si.util.*;
18. import com.ibm.retail.si.Copyright;
19. import com.ibm.retail.AEF.factory.*;
20. import com.ibm.retail.AEF.thread.*;
21. import com.ibm.retail.AEF.data.*;
22. import com.ibm.retail.AEF.session.*;
23. import com.ibm.retail.AEF.action.*;
```

```

24.
25. import java.util.*;
26. import java.rmi.*;
27.
28. import org.apache.commons.logging.Log;
29. import org.apache.commons.logging.LogFactory;
30.
31. /**
32.  * UserStartTransactionActionImpl overrides SStartTransactionActionImpl
33.  * to handle an additional zipcode argument on the start transaction
34.  * key sequence.
35.  *
36.  */
37. public class UserStartTransactionActionImpl extends SStartTransactionActionImpl
38. {
39.
40.     static String copyright()
41.     { return com.ibm.retail.si.Copyright.IBM_COPYRIGHT_SHORT; }
42.     private static AEFPerfTrace perfTrace = AEFPerfTrace.getInstance();
43.
44.     /**
45.      * Constructor
46.      *
47.      * @param request The ActionRequest.
48.      * @exception AEFException
49.      * AEFException error codes:
50.      * <br>AEFConst.INVALID_ARGUMENT, AEFConst.INVALID_TRANSACTION_TYPE
51.      *
52.      */
53.     public UserStartTransactionActionImpl(ActionRequest request) throws
AEFException
54.     {
55.         super(request);
56.
57.         TransactionIdentifier transactionID =
(TransactionIdentifier)(request.getArguments());
58.         zipcode = (String)(transactionID.get("zipcode"));
59.         if (zipcode == null || zipcode.length() == 0)
60.         {
61.             throw new AEFException(AEFConst.INVALID_ARGUMENT,
62.                                     -9000, // -9000 to -9999 reserved for user
error codes
63.                                     "UserStartTransactionActionImpl.constructor():
zipcode is a required argument.");
64.         }
65.     }
66.
67.     /**
68.      * Sends the key sequence which will cause the transaction to be started.
69.      *
70.      * @return int An integer value representing the result of the key sequence.
71.      * @exception AEFException
72.      *
73.      */
74.     public int sendStartTransactionSequence() throws AEFException
75.     {
76.         int retVal = -1;
77.         args.clear();
78.         args.put("SEQUENCE_ID", "startTransaction");
79.         args.put("%0", zipcode);
80.         keySequenceAction = (AEFAction)(actionFactory.makeAction(new
ActionRequest("SimpleKeySequenceAction", args)));
81.         Condition[] goodConditions =
82.         {
83.             new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
84.                                         POSDeviceProperties.POS_SUB_STATE,
85.                                         Substate.getSubstate("SELECT_PROCEDURE")),
86.             new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
87.                                         POSDeviceProperties.POS_SUB_STATE,
88.                                         Substate.getSubstate("TRAINING_MODE")),
89.         };
90.         lock = new ConditionLock();

```

```

91.     detector=((DetectorAccess)(SessionContext.getSession()))).
92.         getTransactionDetector();
93.     // Save any current transaction instance number so we are sure to only get
a transaction
94.     // created after this key sequence is sent.
95.     instanceNumber = detector.getInstanceNumber();
96.
97.     retVal = lock.performActionAndWait("wait-for-expecting-item-entry-
substate",
98.                                     keySequenceAction,
99.                                     goodConditions,
100.                                    BadConditionsImpl.
101.                                        getInstance().getBadConditions(),
102.                                        getTimeout());
103.     if (retVal < 0)
104.     {
105.         AEFErrorHandler errorHandler = new AEFErrorHandler("Send Start
Transaction Sequence Error");
106.         retVal = errorHandler.handleError(lock,
107.                                         keySequenceAction,
108.                                         goodConditions,
109.                                         BadConditionsImpl.
110.                                             getInstance().getBadConditions(),
111.                                             getTimeout(),
112.                                             retVal);
113.     }
114.     return retVal;
115. }
116.
117. /* Instance Variables */
118. protected String                                zipcode;
119. private static Log log =
120.     LoggerFactory.getLog(UserStartTransactionActionImpl.class);
121. }

```

This class overrides the constructor to pull out the zipcode from the arguments as shown on line #58. The sendStartTransactionSequence method is exactly the same as in the parent SASTransactionActionImpl class, with the exception of line #79, which adds the zipcode value to the key sequence to be sent to the application, and the removal of some tracing statements.

After making sure the director **d:\si\userjar** exists, this class may be compiled with the command

```

d:\si\user>javac -classpath
d:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:/
/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar
com/userco/retail/AEF/action/UserStartTransactionActionImpl.java

```

Modifying the Key Sequence

As shown on line #78 in the source code above, the key sequence is defined by the key "startTransaction". We see from "Appseq.properties (SA)" on page 262 that this is currently defined as "0<ENTER>". For this task, we need to redefine this to be "0<SLASH>{0}<ENTER>". The "{0}" is a substitution variable which will be filled in with the zipcode value as shown on line #79 in the source code above. Note that if we needed a second substitution variable, it would be specified as "{1}", and the code to fill in the value would use the key "%1".

To override the key sequence, create or edit the file **d:/si/userjar/appseq.properties**. Add the line

```
startTransaction=0<SLASH>{0}<ENTER>
```

Bundling the Files

The new `UserStartTransactionActionImpl.class` and `userseq.properties` files must be included in the appropriate `user*.jar` file, placed on the 4690, and distributed. This may require CSS and/or the terminals to be stopped.

To create a new `user.jar` file, issue the command

```
d:/si/userjar>jar cfv user.jar *
```

To update an existing `user.jar` file to add the two new files, issue the command

```
d:/si/userjar>jar uvf user.jar  
com/userco/retail/AEF/action/UserStartTransactionActionImpl.class  
userseq.properties
```

Task: Extending the Automation Provider API with New Function

The Automation Provider includes an extension mechanism which applies when none of the existing API's can serve the purpose required by the extension. This mechanism makes use of the "action" model already discussed. In general, a new action class is written which performs the desired function and returns a result. The steps required are:

- Create a new "action" class to perform the new function.
- Configure a class key in `userclass.properties`.
- Package the new class file and properties files in a `user*.jar` file as appropriate.
- Execute the new action by calling the `POSAutomationProvider.performAction` method to execute the new action and return a result.

Creating a New Action Class

The new action class must implement the [AEFAction](#) interface. For example, if we want a new action which performs a gift card activation function, we could create the Java source file

```
d:/si/user/com/userco/retail/AEF/action/GiftCardActivationImpl.java
```

This class must implement the `AEFAction` interface, and define a `performAction` method. If there is a result to be returned to the client, it is returned from the `performAction` method. Note that the object returned must be either `Serializable`, or be an RMI object.

After making sure the director `d:/si/userjar` exists, this class may be compiled with the command

```
d:\si\user>javac -classpath
d:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:
/si/si_jars/siutil.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar
com/userco/retail/AEF/action/GiftCardActivationImpl.java
```

Creating the Class Key

A new class key for the new action class must be added to userclass.properties. Edit (or create) the file `d:/si/userjar/userclass.properties`, and add the line:

```
GiftCardActivationAction=com.userco.retail.AEF.action.GiftCardActi
vationActionImpl
```

Packaging the Files

The new `GiftCardActivationActionImpl.class` and `userclass.properties` files must be included in the appropriate `user*.jar` file, placed on the 4690, and distributed. This may require CSS and/or the terminals to be stopped.

To create a new `user.jar` file, issue the command

```
d:/si/userjar>jar cfv user.jar *
```

To update an existing `user.jar` file to add the two new files, issue the command

```
d:/si/userjar>jar uvf user.jar
com/userco/retail/AEF/action/GiftCardActivationActionImpl.class
userclass.properties
```

Executing the New Action

In order to execute the new action, the client creates an instance of [ActionRequest](#), passing the class key and a `HashMap` of arguments to be used by the action. For this example, assume that the required arguments are a gift card number, and an amount. The client code to execute the `GiftCardActivationActionImpl` would be:

```
// Assume cardNumber and cardAmount are String variables,
// and that automationProvider is a variable pointing to
// a valid POSAutomationProvider instance.

HashMap args = new HashMap();
args.put("cardNumber", cardNumber);
args.put("cardAmount", cardAmount);

// Create a new ActionRequest. The class key for the action
// is the same key we have already put in userclass.properties
// to map to
// com.userco.retail.AEF.action.GiftCardActivationActionImpl.

ActionRequest request = new
    ActionRequest("GiftCardActivationAction", args);

Object result = null;
try
{
    result = automation.performAction(request);
}
catch (Throwable th)
{
    // Error processing here.
```

}

System Management

Task: Writing and Registering Your Own MBeans

There are numerous benefits to enabling application components to be managed, especially in a store environment with numerous, distributed stores. The ability to expose application attributes, allow them to be monitored, manage application configuration, and forward application events to the enterprise can all provide time and cost savings for IT staff. This section describes how to instrument application components with MBeans using JMX, the management platform used by the IBM Store Integrator. The following are the steps involved in performing this task:

- **Exposing the Management Interface** Define the MBean interface and implementation for the managed resource.
- **MBean Object Naming** Define a unique and structured name for the MBean within the management infrastructure. This allows each MBean within the store to be individually searched and accessed from a remote management application.
- **MBean Registration** Register the MBean with the JMX MBeanServer. This exposes the MBean for external access.
- **Distributing the MBean Classes** Install the Java class files associated with the MBean into the correct places within the management infrastructure

Exposing the Management Interface

Making a resource manageable involves the creation of an MBean for that resource. The JMX specification defines multiple types of MBeans in addition to the standard MBean, including dynamic, model, and open MBeans. Even though all are implemented differently and some have additional features for added flexibility, at their core they all expose a management interface for a managed resource. The standard MBean is the simplest type of MBean to implement, requiring only that the managed resource `MyClass` implement the Java interface `MyClassMBean`. This example will use a simple standard MBean called `SampleMBean`, which exposes a read/write String attribute, `Attribute1`, and a read-only boolean attribute, `BoolAttr`. Also exposed is an operation called `operation1`, which takes a single String parameter and returns an integer. The `SampleMBean` interface is the MBean interface which is implemented by the managed resource, `Sample`.

SampleMBean:

```
public interface SampleMBean {  
    public String getAttribute1();  
    public void setAttribute1( String attr1Val );  
    public boolean isBoolAttr();  
    public int operation1( String param );  
}
```

MBean Object Naming

Within the scope of a JMX manageable system, an MBean derives its unique identity not from the instance itself, as this can not be scoped across multiple JVMs, but rather from a unique characteristic of an MBean – its `ObjectName`. An MBean's `ObjectName` can be thought of simply as collection of comma separated “Key=value” pairs.

For Example: “Domain:name=name,foo=one,bar=two,more=three”

Unfortunately, JMX does not provide any guidance or general rules for the application of these names. That is left entirely in the space of MBean provider. In order to provide some organization of MBeans within the Store Integrator management infrastructure, some rules and conventions that should be applied and followed when naming MBeans within and across the Store Integration Framework product space.

The real value in `ObjectNames` is in deriving some sort of contextual usage from them. Specifically, an application that consumes these MBeans may benefit from having a known structured format with specific content. This is the case for the Store Integration Framework, but more importantly, the intent is to provide not only context, but a structured space that will eliminate the possibilities of naming collisions.

The name space specified by this infrastructure is segmented into three parts:

1. System-Wide Identification

These system-wide identification elements are always obtainable from the General/Master Agent in a SIF deployment, using the singleton instance of [ObjectNameFactory](#) provided in the SIF mgmt package. Using the

`createObjectName` methods of this class, it is possible to build a name that is consistent with the SIF requirements where these elements are automatically added.

- a. SIF MBean Identifier
This identifies an MBean as being related to SIF.
Syntax: **SIFMBean=true**
- b. Store ID
This is retrieved as part of the Master / General Agent relationship, and is unique to a given store. It is only applied to proxied MBeans on the Master Agent. It is a configured element on the Master Agent.
Syntax: **StoreID=x**
- c. Device ID
This is an automatically generated element of the General Agent, and is unique to each physical device within the store. This value will default to the device's hostname and then to an integer representation of the device's IP address. For 4690 terminals, it will be <controller id>.< terminal number>, and for controllers, just <controller id>.
Syntax: **DeviceID=x**
- d. System ID
This is an automatically generated element of the General Agent, and is unique to each agent running in the store. The system Id takes the form: <deviceId>.<mgmtPort>, where mgmtPort is the agent's management port.
Syntax: **SystemId=<deviceId>.<mgmtPort>**

2. Component Identification

- a. Component Name
Unique identification for a functional component (Required). In the case of the Store Integrator components, there are a list of reserved component names.
Syntax: **SIFComponent=xyz**
- b. Device Major Version
The Major version of the component being represented (Optional).
Syntax: **DMajorVer=x**
- c. Device Minor Version
The Minor version of the component being represented (Optional)
Syntax: **DMinorVer=x**

3. Device Content (Component/Application specific data)

- a. Type Identifier
Uniquely identifies the MBean within the component (Required). For MBeans where only one instance exists in the system, this would be the MBean type (i.e. SessionManager). For MBean types that can have multiple instances, an additional tag should be added to the type (i.e. Session.200).
Syntax: **Id=<Id>**

There are no requirements for the domain portion of the `ObjectName`, since it is sometimes externally controlled. The `SIFMBean` Identifier key will be used to identify SIF MBeans.

Putting all of the pieces together, a component would build its `ObjectName` by first constructing the component specific portion and then acquiring the System-Wide information from the General/Master Agent it is working with by executing the following:

```
MgmtAgent generalAgent = MgmtAgentFactory.getGeneralAgent();
```

```

ObjectName newName = new ObjectName(mbs.getDefaultDomain() +
":type=Sample" );
ObjectNameFactory Namer = ObjectNameFactory getInstance();
try{
    newName = Namer.createObjectName(newName,
                                     sample1,
                                     "AEF",
                                     null,
                                     null );
} catch (MalformedObjectNameException err){...}

```

The result is an ObjectName that looks like something like the following:

```

generalagent:SifMBean=true,StoreID=1,DeviceID=1,SystemId=1.10151,S
IFComponent=AEF,type=sample1,id=sample1

```

MBean Registration

All MBeans are nothing but Java objects until they are exposed through the JMX MBeanServer through the `registerMBean` or `createMBean` methods. Each SI Management GeneralAgent creates and manages a MBeanServer through which MBeans can be added to the Store Integration Framework infrastructure. The method [getMBeanServer](#), defined in the MgmtAgent interface implemented by the GeneralAgent class, returns this MBeanServer instance. The MgmtAgentFactory class is the point of access to the singleton GeneralAgent instance in each JVM. An example of how one might obtain the MBeanServer and register an instance of the SampleMBean defined previously is as follows:

```

MgmtAgent genAgent = null;
int deviceType = MgmtConst.dType4690;
MBeanServer mbs = null;

synchronized( MgmtAgentFactory.class ) {
    try {
        if( MgmtAgentFactory.isGeneralAgentRunning() )
            genAgent = MgmtAgentFactory.getGeneralAgent();
        else
            genAgent = MgmtAgentFactory.getGeneralAgent( deviceType );
        mbs = genAgent.getMBeanServer();
    } catch ( MgmtException me ) { ... }
}

try {
    Sample resource = new Sample();
    Mbs.registerMBean( resource, newName );
} catch ( Exception e ) { ... }

```

Distributing the MBean Classes

For the registered MBean to be managed across the SIF infrastructure, the Java classes that comprise the MBean must be in the classpaths at each point in the infrastructure. This includes in the JVM where the General Agent is running, the Master Agent on the in store processor, and in the Store Integrator Viewer web application.

A convention exists for adding additional classes to the classpath of the Master Agent. Any jar files located in the directory `$COMMON_HOME/simgmt/ext` are automatically

added the Master Agent classpath during startup. When new classes need to be added, placing the jar files in this directory and restarting the SIF Management Service is all that is required. Adding classes to the classpath of the Store Integrator Viewer involves copying a jar file with the classes to the WEB-INF/lib directory of the application, and restarting the application.

Task: Interfacing with the MgmtNotificationControlMBean:

The `MgmtNotificationControlMBean` is a central collection point on the Master Agent for JMX Notifications emitted by each General Agent. The MBean resends all of the Notifications that it receives and also temporarily stores those of filtered types in a rolling queue for later access. This section describes the two ways in which to receive or access Notification data:

- **Real Time:** Add a `NotificationListener` to receive Notifications in real time
- **Enumerate Stored Notifications:** Use the `MgmtNotificationControlMBean` management interface to enumerate the stored notifications received while offline

Real Time Notifications

Receiving general agent Notifications as they are received and resent by the `MgmtNotificationControlMBean` is as simple as registering an `NotificationListener` to it via one of the JMX MBeanServer's `addNotificationListener` methods (observer is an `ObjectName` or a plain Java Object that implements the `NotificationListener` interface):

```
ObjectName notifControlName = new ObjectName(
    MgmtNotificationCtrlMBean.OBJECT_NAME );
connection.addNotificationListener( notifControlName, observer,
    null, null );
```

Enumerate Stored Notifications

As Notifications are received by the Notification Control, they are persisted in a rolling queue (The size of which is configurable). Operations defined by the `MgmtNotificationControlMBean` interface allow management applications to enumerate through the current set of stored Notifications. The `getFirstStoredNotification` operation returns the oldest stored Notification and `getNextStoredNotification` returns subsequent Notifications. The sequence number of the current notification must be passed to the `getNextStoredNotification` operation, which will return the next Notification in the sequence. The following is an example of how one would enumerate through all stored Notifications:

```
Notification notification = notCtrl.getFirstStoredNotification();
while( notification != null ) {
    // Do something with notification
    notification = getNextStoredNotification(
        notification.getSequenceNumber() );
}
```

Note: The same requirements for distributing MBean classes apply to user defined Notification classes.

Coding Guidelines

Task: Using Error Logging

Error logs are critical in detecting and correcting application errors. The more information about what the application was doing at the time of the problem, the easier it is to figure out what the problem is. Unfortunately, logging too much information can have a negative impact on performance. Therefore, finding the right amount of information to log is very important so that problems can be resolved without impacting performance. This task will show you how to log errors in a way that is consistent with the rest of the Store Integrator code. Topics such as error log entry contents, APIs, and correct use of logging levels will be discussed.

Logging Levels

Because of the negative impact of logging too much information on the system, it is important to use logging levels consistently. This allows you to know what kind of information is logged at each level. Using this knowledge, you can set the logging level for your system to give you just the information that you need. The following table shows what each logging level should be used for.

FATAL	Severe errors that cause premature termination. Expect these to be immediately visible on a status console
ERROR	Other runtime errors or unexpected conditions. Expect these to be immediately visible on a status console. This level is used for: <ul style="list-style-type: none">• Java Exceptions• Missing data files such as EALAEFXF.DAT or EAMAEFXF.DAT• Missing property files
WARN	Use of deprecated APIs, poor use of API, 'almost' errors, other runtime situations that are undesirable or unexpected, but not necessarily "wrong". Expect these to be immediately visible on a status console. This level is used for: <ul style="list-style-type: none">• Missing bundle keys or values.• Factory cannot create object messages.• Invalid values.• Condition lock timeouts which will include condition states.
INFO	Interesting runtime events (startup/shutdown). Expect these to be immediately visible on a console, so be conservative and keep to a minimum. This level is used for: <ul style="list-style-type: none">• Error Helper creation.• Error Helper handleError() return values.• AEFBase configuration.• Dump of the Bundles which include all the values from the property files.• Java Health Pipe creation.
DEBUG	Detailed information on flow of through the system. Expect these to be written to logs only. This level is used for: <ul style="list-style-type: none">• Automation call summary (e.g. Operator 999999 logon called)• SalesTransaction call summary (e.g. Add item 3 (itemid toString())called).• XML statements received from base application.

TRACE	<p>More detailed information. Expect these to be written to logs only. By default the message priority should be no lower than info. That is, by default debug and trace messages should not be seen in the logs. You want to have exception/problem information available for first-pass problem determination in a production level enterprise application without turning on debug or trace as default log levels. There is simply too much information in these levels to be appropriate for day-to-day operations.</p> <p>This level is used for:</p> <ul style="list-style-type: none"> Method entry and exit tracing.
--------------	---

Table 12 Logging Levels

Logging APIs

The following table shows the list of APIs and their affect after the Jakarta Commons and Sun's JDK 1.4 differences are factored in. So in effect these are the APIs that Store Integrator uses. It is probably worth pointing out that the idea behind the Jakarta Commons Logging Wrapper is to be logging implementation independent. So you should treat `log.fatal()` and `log.error()` differently even though they will look the same in the error logs. That way if your logging implementation changes to one that distinguishes between error and fatal levels, your software will take advantage of that.

API	Log Level	Sample Log Entry
Log.fatal() Log.error()	SEVERE	Apr 21, 2003 8:10:48 AM Test main SEVERE: error text Apr 21, 2003 8:10:48 AM Test main SEVERE: error text
Log.warn()	WARNING	Apr 21, 2003 8:10:48 AM Test main WARNING: error text
Log.info()	INFO	Apr 21, 2003 8:10:48 AM Test main INFO: error text
Log.debug()	FINE	Apr 21, 2003 8:10:48 AM Test main FINE: error text
Log.trace()	FINEST	Apr 21, 2003 8:10:48 AM Test main FINEST: error text

Table 13 Log Levels with Sample Log Entries

Controlling Error Log Output

The output of the logging calls is configurable based on the settings of the `logging.properties` file specified when the JVM was started. All of the Store Integrator response files include an entry for a logging properties file to use (-Djava.util.logging.config.file=ADX_IPGM:cmxlogs.pro). Please see the *Store Integrator User's Guide* for more information about controlling the output of the logging calls. What is important to understand here is that the end user has the ability to control what information makes it into the error logs. Since logging an error involves some overhead, it is recommended that a test be done before an error is logged. This way if the error log is just going to be thrown away, the overhead is avoided. Specifically, it is recommended that `log.isInfoEnabled()`, `log.isDebugEnabled()`, and `log.isTraceEnabled()` be called before logging an error at those levels. For example:

```
if log.isInfoEnabled()
{
    log.info("Text to be logged");
}
```

Error Log Entry Contents

There are two versions of the log APIs for each logging level. One that takes a string and one that takes a string and an exception as a parameter. In order to standardize the log entries, it is recommended that any errors that include exceptions, should be logged using the two parameter version of the API. Converting the Exception objects into a string will lead to inconsistent looking log entries. To further standardize the log entries, Store Integrator provides a class called [AEFMessage](#). This class has two attributes. The first is the message which contains the text part of the error. The second is sessionId which should be set to the terminal number of the terminal logging the error. These attributes should be set and this class should be passed as the string parameter of the logging calls. Here is an example of logging using the `AEFMessage` class:

```
msg = new AEFMessage();
if (log.isDebugEnabled())
{
    msg.setSessionID(terminalNumber);
    msg.setMessage("Creating ApplicationDataConnector.");
    log.debug(msg);
}
```

Sample Error Logging Program

The following program shows how the logging code should be used. The first three lines are the import statements needed for `AEFMessage` and the logging classes. In line 6 the `Log` object is created. It is recommended that the `Log` object have the same name as the class. So in this case we pass `Test.class` to the `getLog` method. On line 9 a new `AEFMessage` object is created. The terminal number for this `AEFMessage` object is initialized on line 10. Lines 11, 12, and 15 are used to insure that the logging calls are not made if this level of logging is not enabled for this log object (commonly referred to as a code guard). Line 13 initializes the message attribute of the `AEFMessage` object. In line 14 a trace message is logged. Lines 16 through 20 are the same as 11 through 15 except that in this case a debug message is logged. Lines 21 through 33 show how to log warning and fatal messages. As you can see no code guard was used for these calls because the errors are higher than informational. Lines 34 through 38 log a trace message to match the one from lines 11 through 15.

```
1. import com.ibm.retail.AEF.util.*;
2. import org.apache.commons.logging.Log;
3. import org.apache.commons.logging.LogFactory;
4. public class Test
5. {
6.     private static Log log = LogFactory.getLog(Test.class);
7.     public static void main(String[] args) throws Exception
8.     {
9.         AEFMessage msg = new AEFMessage();
10.        msg.setSessionID("001");
11.        if (log.isTraceEnabled())
12.        {
13.            msg.setMessage("Entering Test.main().");
14.            log.trace(msg);
15.        }
16.        if (log.isDebugEnabled())
17.        {
18.            msg.setMessage("Test.main() called with the following arguments:
19.            " + args);
20.            log.debug(msg);
21.        }
22.        try
23.        {
24.            if (args.length == 0)
```

```

25.         msg.setMessage("Main called with no arguments. Defaults will
    be used.");
26.         log.warn(msg);
27.     }
28. }
29. catch (Exception e)
30. {
31.     msg.setMessage("An exception was thrown in the main method of
Test.");
32.     log.fatal(msg, e);
33. }
34. if (log.isTraceEnabled())
35. {
36.     msg.setMessage("Leaving Test.main().");
37.     log.trace(msg);
38. }
39. }
40. }

```

Task: Writing an Error Helper

An error helper is a class that is called by the [AEFErrorHandler](#) class to handle errors or provide input to the base application. The idea is for each error helper to handle a specific error or input sequence. If the error or condition is corrected, the error helper returns a value of `ErrorHelper.ERROR_HANDLED` to the [AEFErrorHandler](#). This includes a case where the original error or condition is cleared but another error is detected. If the error or condition is not corrected, then the error helper will return `ErrorHelper.ERROR_IS_FATAL` to the [AEFErrorHandler](#).

Error Helper Classes

All the error helpers must implement the [ErrorHelper](#) interface. Store Integrator provides several abstract implementations of this interface. These include `ErrorHelperImpl`, `SAErrorHelper`, `ACEErrorHelper`, and `SAErrorHelper`. Store Integrator also provides a default error helper that gets called if an error is encountered that has no error helper configured for it. This default error helper is `DefaultErrorHelper`. This error helper always returns `ErrorHelper.ERROR_IS_FATAL` and sets its outer exception to indicate that the default error helper was used.

ErrorHelper Interface Methods

The following methods are specified in the [ErrorHelper](#) interface.

1. `getError` - Used to get the `AEFError` object that contains the information about the error this helper is trying to handle. `ErrorHelperImpl` provides an implementation of this method.
2. `setError` - Sets the error that this helper is working on. `ErrorHelperImpl` provides an implementation of this method.
3. `getOuterException` - Returns the outer most Exception (if any) that was thrown while handling this error. `ErrorHelperImpl` provides an implementation of this method.
4. `setOuterException` - Sets the outer most Exception that was thrown while handling this error. `ErrorHelperImpl` provides an implementation of this method.
5. `getSleepInterval` - Returns the number of milliseconds to wait before continuing with the next operation while handling errors. During testing, it was found that differences in hardware, and software configuration could lead to timing issues. In many cases, invalid key sequence errors were erroneously generated. In order to compensate for this, Java sleep calls were added to the `AEFErrorHandler` and `ErrorHelper` classes. The Sleep Interval is configured in `config.properties`. All of these timing issues have been corrected but this method was left here in case it is needed in the future. `ErrorHelperImpl` provides an implementation of this method.
6. `sleep` - Causes the Error Helper to wait before continuing with the next operation. During testing, we found that differences in hardware, and software configuration could lead to timing issues. In many cases, invalid key sequence errors were erroneously generated. In order to compensate for this, sleep calls (using this method) were added to the `AEFErrorHandler` and `ErrorHelper` classes. The default sleep interval is configured in `config.properties`. All of these timing issues have been corrected but this method was left here in case it is needed in the future. The default value for the sleep interval was set to -1 to avoid the call to the `Thread.sleep` method. `ErrorHelperImpl` provides an implementation of this method.

7. `isEmulatedDevice` - Used to determine if a device is real or emulated. `ErrorHelperImpl` provides an implementation of this method.
8. `getCurrentState` - Gets the current state of the base application. `ErrorHelperImpl` provides an implementation of this method.
9. `setInputDevicesLocked` - Locks the input devices. `ErrorHelperImpl` provides an implementation of this method.
10. `resetInputDevices` - Reset the input devices to the state prior to the `setInputDevicesLocked` call. `ErrorHelperImpl` provides an implementation of this method.
11. `getErrorHandler` - Gets the `AEFErrorHandler` that created this error helper. `ErrorHelperImpl` provides an implementation of this method.
12. `setErrorHandler` - Sets the `AEFErrorHandler` that created this error helper. `ErrorHelperImpl` provides an implementation of this method.
13. `handleError` - This method is the one that does the work specific to clearing this error. Each error helper must implement this method.

Error Helper Bundles

The error and class bundles provide information that the AEF error handler needs to create the error helpers. From the error bundle, the error helper gets:

1. A text description of the error.
2. The key (from the class bundle) of the error helper to use to handle the error. (if a class name is provided the system will create that class but it is better to provide a key instead).
3. The error code associated with this error. This code will be extracted from the `AEFConst.java` file. If instead of a code a number is used, the error helper will use that number instead. The numbers from 9000 to 9999, and -9000 to -9999 are reserved for user programming.
4. The extended error code associated with this error. This code will be extracted from the `AEFConst.java` file. If instead of a code a number is used, the error helper will use that number instead. The numbers from 9000 to 9999, and -9000 to -9999 are reserved for user programming.

From the class bundle, the error helper gets the class to use to create the error helper. The correct place to make changes to these bundles is the `userclass.properties`, and `usererror.properties` files. These files should be included in one of the user jar files so that the application will have access to them at runtime.

Sample Informational Error Helper

The following code shows a simple error helper. This is the SA error helper that is used for informational error messages. These are messages where the operator can acknowledge the message but is not given any other choices. Lines 1 through 11 are the import statements needed for a typical error helper. On line 13 the class declaration extends `SAErrorHandler`. `SAErrorHandler` extends `ErrorHelperImpl` so most of the methods of the `ErrorHelper` interface have been implemented already. In lines 33 through 117 the `handleError` method is implemented. It is worth pointing out here that there are three error handling modes. Of these, only two are supported today. These are "automatic" (`POSAutomationProvider.HANDLE_AUTOMATIC`), and "default" (`POSAutomationProvider.HANDLE_DEFAULT`). The automatic mode is used to let Store Integrator attempt to clear the errors without operator intervention. This is helpful for customer facing devices where the consumer should not be asked to clear errors or enter manager override codes, etc. In the default mode, Store Integrator waits for the operator to clear the error or input the data then continues with the action. This is useful for devices that are operated by store personnel.

```

1. package com.ibm.retail.AEF.util;
2. import com.ibm.retail.AEF.action.*;
3. import com.ibm.retail.AEF.automation.*;
4. import com.ibm.retail.si.util.*;
5. import com.ibm.retail.si.Copyright;
6. import com.ibm.retail.AEF.thread.*;
7. import com.ibm.retail.AEF.session.*;
8. import java.rmi.*;
9.
10. import org.apache.commons.logging.Log;
11. import org.apache.commons.logging.LogFactory;
12.
13. public class SAErrorHandlerInformational extends SAErrorHandler
14. {
15.
16.     // Constructor
17.     public SAErrorHandlerInformational() throws AEFException
18.     {
19.
20.
21.     /**
22.      * Tries to resolve the error condition if possible.
23.      * @param lock The ConditionLock that returned the error.
24.      * @param keySequenceAction The key sequence that generated the error.
25.      * @param goodConditions An array of the conditions that would indicate
26.      * success after the key sequence is sent.
27.      * @param badConditions An array of the conditions that would indicate a
28.      * failure after the key sequence is sent.
29.      * @param timeout An integer used to determine how long to wait for one of the
30.      * listed conditions before a timeout exception is thrown.
31.      * @param satisfiedCondition The index of the condition in the badConditions
32.      * array that was satisfied (needs to be adjusted as it is a negative number).
33.      * @return int Returns ErrorHandler.ERROR_HANDLED if the error was cleared or
34.      * dealt with.
35.      * Returns ErrorHandler.ERROR_IS_FATAL if the error was not cleared or dealt
36.      * with.
37.      */
38.
39.     public int handleError(ConditionLock lock, AEFAction keySequenceAction,
40.         Condition[] goodConditions, Condition[] badConditions, int timeout, int
41.         satisfiedCondition)
42.     {
43.         int retVal = 0;
44.         int handlingMode = 0;
45.         try
46.         {
47.             handlingMode = getErrorHandlingMode();
48.             switch(handlingMode)
49.             {
50.                 case POSAutomationProvider.HANDLE_AUTOMATIC:
51.                     sleep(); // Added to resolve timing issues. See sleep method
52.                     // for more info.
53.                     retVal = clearError();
54.                     break;
55.                 case POSAutomationProvider.HANDLE_DEFAULT:
56.                     // Unlock keyboard if need be so that the operator can
57.                     // interact.
58.                     setInputDevicesLocked(false);
59.                     retVal = waitForOperatorToClear();
60.                     break;
61.                 case POSAutomationProvider.HANDLE_CALLBACK:
62.                     // Not Implemented in this release!
63.                 default:
64.                     throw new
65.                         AEFException(AEFConst.UNSUPPORTED_ERROR_HANDLING_MODE,
66.                             AEFConst.NONE, "An Error Handling Mode of " + handlingMode +
67.                             " was returned which is not valid.");
68.             }
69.         }
70.         catch (RemoteException re)
71.         {

```

```

59.         AEFException tmpException = new
            AEFException(AEFConst.ERROR_HANDLER_EXCEPTION_DETECTED,
            AEFConst.RMI_REMOTE_EXCEPTION,
            "SAErrorHelperInformational.handleError(): Remote Exception was
            detected.", re);
60.         tmpException.appendExceptions(getOuterException());
61.         setOuterException(tmpException);
62.         retVal = ErrorHandler.ERROR_IS_FATAL;
63.         if (log.isWarnEnabled())
64.         {
65.             tempAEFMessage.setMessage("There was a remote exception thrown in
                handleError of SAErrorHelperInformational for error " +
                getError().getErrorKey() + ".");
66.             log.warn(tempAEFMessage, re);
67.         }
68.     }
69.     catch (AEFException e)
70.     {
71.         e.appendExceptions(getOuterException());
72.         setOuterException(e);
73.         retVal = ErrorHandler.ERROR_IS_FATAL;
74.         if (log.isWarnEnabled())
75.         {
76.             tempAEFMessage.setMessage("There was an AEF exception thrown in
                handleError of SAErrorHelperInformational for error " +
                getError().getErrorKey() + ".");
77.             log.warn(tempAEFMessage, e);
78.         }
79.     }
80.     finally
81.     {
82.         // Reset the input devices.
83.         try
84.         {
85.             resetInputDevices();
86.         }
87.         catch (AEFException e)
88.         {
89.             e.appendExceptions(getOuterException());
90.             setOuterException(e);
91.             retVal = ErrorHandler.ERROR_IS_FATAL;
92.             if (log.isWarnEnabled())
93.             {
94.                 tempAEFMessage.setMessage("There was an AEF exception thrown in
                    handleError of SAErrorHelperInformational for error " +
                    getError().getErrorKey() + ".");
95.                 log.warn(tempAEFMessage, e);
96.             }
97.         }
98.     }
99.     if (retVal < 0)
100.    {
101.        retVal = ErrorHandler.ERROR_IS_FATAL;
102.    }
103.    else
104.    {
105.        retVal = ErrorHandler.ERROR_HANDLED;
106.    }
107.    if (log.isDebugEnabled())
108.    {
109.        tempAEFMessage.setMessage("Leaving handleError for error " +
            getError().getErrorKey() + ".");
110.        log.debug(tempAEFMessage);
111.        tempAEFMessage.setMessage("The return code was " + retVal + ".");
112.        log.debug(tempAEFMessage);
113.        tempAEFMessage.setMessage("The Error Handling mode used was " +
            handlingMode + ".");
114.        log.debug(tempAEFMessage);
115.    }
116.    return retVal;
117. }
118.

```

```

119. // Instance Variables
120. private static Log log =
    LoggerFactory.getLog(SAErrorHelperInformational.class);
121. }

```

Writing your Own Error Helper

This task assumes that you are running the SA application. If you are not running the SA application, you can still follow along, but you will not be able to run the resulting code.

In this task you are going to write an error helper that will notify store personnel when a “B082 PICKUP NEEDED SOON” prompt is displayed by a terminal. By default B082 errors in the Supermarket application are handled by the `SAErrorHelperInformational` class (shown above). So in automatic mode the error is cleared by the system. In default mode the error is cleared by the operator. In neither case does the system log an error. If the above error helper was modified to log an error, then the Systems Management capabilities built into Store Integrator would allow you to monitor for this error and take preventive actions. Please see the Systems Management section in this guide for more information on monitoring errors.

Here are the steps needed to accomplish this task.

1. Write the new error helper.
2. Determine what error codes to use.
3. Modify the Error bundle.
4. Modify the Class bundle.
5. Include all the new and modified files in a user jar file.
6. Install the user.jar file on the system.

Write the New Error Helper

The following code shows the new error helper. This java file name and path should be **d:/si/user/com/userco/retail/AEF/util/PickupNeededSoonErrorHelper.java**. This is the error helper (discussed above) that is used for informational error messages with a few new lines added. These lines (41 through 45) are used to log an error message.

```

1. package com.userco.retail.AEF.util;
2. import com.ibm.retail.AEF.action.*;
3. import com.ibm.retail.AEF.automation.*;
4. import com.ibm.retail.si.util.*;
5. import com.ibm.retail.si.Copyright;
6. import com.ibm.retail.AEF.thread.*;
7. import com.ibm.retail.AEF.session.*;
8. import com.ibm.retail.AEF.util.*;
9. import java.rmi.*;
10.
11. import org.apache.commons.logging.Log;
12. import org.apache.commons.logging.LogFactory;
13.
14. public class PickupNeededSoonErrorHelper extends SAErrorHelper
15. {
16.
17.     // Constructor
18.     public PickupNeededSoonErrorHelper() throws AEFException
19.     {
20.     }
21.
22.     /**
23.      * Tries to resolve the error condition if possible.
24.      * @param lock The ConditionLock that returned the error.
25.      * @param keySequenceAction The key sequence that generated the error.
26.      * @param goodConditions An array of the conditions that would indicate
        success after the key sequence is sent.

```

```

27.  * @param badConditions An array of the conditions that would indicate a
28.      failure after the key sequence is sent.
29.  * @param timeout An integer used to determine how long to wait for one of
30.      the listed conditions before a timeout exception is thrown.
31.  * @param satisfiedCondition The index of the condition in the badConditions
32.      array that was satisfied (needs to be adjusted as it is a negative
33.      number).
34.  * @return int Returns ErrorHandler.ERROR_HANDLED if the error was cleared
35.      or dealt with.
36.  * Returns ErrorHandler.ERROR_IS_FATAL if the error was not cleared or dealt
37.      with.
38.  */
39.
40.  public int handleError(ConditionLock lock, AEFAction keySequenceAction,
41.      Condition[] goodConditions, Condition[] badConditions, int timeout,
42.      int satisfiedCondition)
43.  {
44.      int retVal = 0;
45.      int handlingMode = 0;
46.      try
47.      {
48.          handlingMode = getErrorHandlingMode();
49.          if (log.isInfoEnabled())
50.          {
51.              tempAEFMessage.setMessage("Pickup Needed Soon!");
52.              log.info(tempAEFMessage);
53.          }
54.          switch(handlingMode)
55.          {
56.              case POSAutomationProvider.HANDLE_AUTOMATIC:
57.                  sleep(); // Added to resolve timing issues. See sleep method for
58.                      more info.
59.                  retVal = clearError();
60.                  break;
61.              case POSAutomationProvider.HANDLE_DEFAULT:
62.                  // Unlock keyboard if need be so that the operator can interact.
63.                  setInputDevicesLocked(false);
64.                  retVal = waitForOperatorToClear();
65.                  break;
66.              case POSAutomationProvider.HANDLE_CALLBACK:
67.                  // Not Implemented in this release!
68.                  default:
69.                      throw new AEFException(AEFConst.UNSUPPORTED_ERROR_HANDLING_MODE,
70.                          AEFCnst.NONE, "An Error Handling Mode of " + handlingMode +
71.                          " was returned which is not valid.");
72.          }
73.      }
74.      catch (AEFException e)
75.      {
76.          e.appendExceptions(getOuterException());
77.          setOuterException(e);
78.          retVal = ErrorHandler.ERROR_IS_FATAL;
79.          if (log.isWarnEnabled())
80.          {
81.              tempAEFMessage.setMessage("There was an AEF exception thrown in
82.                  handleError of PickupNeededSoonErrorHandler for error " +
83.                  getError().getErrorKey() + ".");
84.              log.warn(tempAEFMessage, e);
85.          }
86.      }
87.      finally
88.      {
89.          // Reset the input devices.
90.          try
91.          {
92.              resetInputDevices();
93.          }
94.          catch (AEFException e)
95.          {
96.              e.appendExceptions(getOuterException());
97.              setOuterException(e);
98.              retVal = ErrorHandler.ERROR_IS_FATAL;
99.          }
100.      }

```

```

86.         if (log.isWarnEnabled())
87.         {
88.             tempAEFMessage.setMessage("There was an AEF exception thrown in
                handleError of PickupNeededSoonErrorHandler for error " +
                getError().getErrorKey()+ ".");
89.             log.warn(tempAEFMessage, e);
90.         }
91.     }
92. }
93. if (retVal < 0)
94. {
95.     retVal = ErrorHandler.ERROR_IS_FATAL;
96. }
97. else
98. {
99.     retVal = ErrorHandler.ERROR_HANDLED;
100. }
101. if (log.isDebugEnabled())
102. {
103.     tempAEFMessage.setMessage("Leaving handleError for error " +
        getError().getErrorKey()+ ".");
104.     log.debug(tempAEFMessage);
105.     tempAEFMessage.setMessage("The return code was " + retVal + ".");
106.     log.debug(tempAEFMessage);
107.     tempAEFMessage.setMessage("The Error Handling mode used was " +
        handlingMode + ".");
108.     log.debug(tempAEFMessage);
109. }
110. return retVal;
111. }
112.
113. // Instance Variables
114. private static Log log =
    LoggerFactory.getLog(PickupNeededSoonErrorHandler.class);
115. }

```

The destination directory will be c:/si/userjar. You need to make sure this directory exists, then compile the file by executing the command

```

C:\si\user>javac -classpath d:/si/si_jars/aef.jar;d:/si/si_jars/aefsa.jar;d:/si/
si_jars/siutil.jar;d:/si/si_jars/c_logging.jar -d d:/si/userjar com/userco/retail
/AEF/util/PickupNeededSoonErrorHandler.java

```

This new class needs to be compiled and the PickupNeededSoonErrorHandler.class from this step will be included into the user.jar later on in this section.

Determine What Error Codes to Use

In the AEFConst.java file, there is a group of public static final integer variables that should be used to represent the error codes associated with a specific error. There is also a group of public static final integer variables that should be used to represent the extended error codes associated with a specific error. Sometimes you will be able to reuse these for your errors. Other times, you will have to use one of the numbers reserved for user errors. In this case we will create a new error code and reuse an extended error code. For the error code you will use 9000. For the extended error code we will use the variable PICKUP_NEEDED. We will use these values in the Error Bundle section below.

Modify the Error Bundle

You must either create a new c:/si/userjar/usererror.properties file or modify an existing one. Once you have the file, add or modify the lines for error B082 as needed to match the lines below.

```
# B082 Pickup Needed Soon
```

```
B082_ADDITIONAL_TEXT=The first cash drawer limit has been exceeded please do a pickup.
```

```
B082_HANDLING_CLASS_KEY= PickupNeededSoonErrorHandler
```

```
B082_ERROR_CODE=9000
```

```
B082_EXTENDED_ERROR_CODE= PICKUP_NEEDED
```

Modify the Class Bundle

You must either create a new `c:/si/userjar/userclass.properties` file or modify an existing one. Once you have the file, add the following line to the file.

```
PickupNeededSoonErrorHandler=com.userco.retail.AEF.util.PickupNeededSoonErrorHandler
```

Include All the New and Modified Files in a User Jar File

In this example we modified or created three files. These files are `c:/si/userjar/usererror.properties`, `c:/si/userjar/userclass.properties`, and `c:/si/userjar/com/userco/retail/AEF/util/PickupNeededSoonErrorHandler.class` (from `PickupNeededSoonErrorHandler.java`). To create a new `user.jar` file that includes all of these files, just invoke the command

```
d:/si/userjar>jar cvf user.jar *
```

from the `userjar` directory. To add the files to an existing file invoke the `jar uvf user.jar *.*` command from the `userjar` directory. Just make sure the `user.jar` file is in the `userjar` directory.

Install the user.jar File on the System

The response files that are shipped with Store Integrator include `c:\user\user.jar` in the classpath. So the new `user.jar` should be put into that directory and distributed to all the controllers. If Java TOF is being used with AEF only, then `c:\user\user.jar` must be added to `ADX_IPGM:aef.duj`, and `ADX_IPGM:aef.rci`. Then `adx_ipgm:aeftof.bat` must be run. If Java TOF is being used with SI GUI, then `c:\user\user.jar` must be added to `ADX_IPGM:sureview.duj`, and `ADX_IPGM:sureview.rci`. Then `adx_ipgm:tof.bat` must be run. After the terminals are reloaded, the new error handler will be invoked when the B082 error is encountered.

Task: Handling Exceptions

When Java encounters an error during program execution, it throws an exception. These exceptions are a subclass of the Java `Throwable` class. The Store Integrator API is designed to throw [AEFException](#) and `RemoteException` objects. The following sections provide more detailed information about these types of exceptions and how to deal with them.

AEFExceptions

`AEFException` extends the standard Java `Exception` class. This class is used to add additional information to exceptions that is unique to Store Integrator.

`AEFException` has the following protected attributes.

- `int errorCode` (Defaults to 0. See `AEFConst.java` for more information.)
- `int errorCodeExtended` (Defaults to 0. See `AEFConst.java` for more information.)
- `String errorKey` (Defaults to null)
- `Throwable originalThrowable`;
- `AEFException previousException` (Defaults to null)
- `AEFException nextException` (Defaults to null)
- `AEFError errorObject` (Defaults to null)

`AEFException` has the following public methods.

- `int getErrorCode()` (Returns the error code for the exception.)
- `void setErrorCode(int newErrorCode)` (Sets the error code for this exception.)
- `int getErrorCodeExtended()` (Returns the extended error code for the exception.)
- `void setExtendedErrorCode(int newErrorCode)` (Sets the extended error code for this exception.)
- `String getDescription()` (Returns the description for the exception.)
- `String getErrorKey()` (Returns the `AEFErrorKey` for this exception.)
- `void setErrorKey(String newErrorKey)` (Sets the `AEFErrorKey` for this exception.)
- `AEFException getPreviousException()` (Gets the previous `AEFException` on the list of chained exceptions.)
- `void setPreviousException(AEFException prevException)` (Sets the previous `AEFException` on the list of chained exceptions.)
- `AEFException getNextException()` (Gets the next `AEFException` on the list of chained exceptions.)
- `void setNextException(AEFException newNextException)` (Set the next `AEFException` on the list of chained exceptions.)
- `Throwable getOriginalThrowable()` (Gets the original throwable (the cause in JVM 1.4 >) for this exception.)
- `void setOriginalThrowable(Throwable newOriginalThrowable)` (Set the original throwable (the cause in JVM 1.4 >) for this exception.)
- `void appendExceptions(AEFException exceptionsToAppend)` (This method takes an `AEFException` and sets the Previous Exception attribute for the last `AEFException` object on this list (the object that we called the method on) to point to it. The idea is to have this (the object that we called the method on) object (the newest exception) be at the start of the list of exception. Traversing the list of Previous Exception attributes would lead you back in time until the oldest exception in the chain is reached.)
- `void printExceptions()` (This method will print all the exceptions in the chain starting with this exception.)

- `String getStackText(Throwable currentThrowable, int indentLevel)` (This method will return the stack trace for a throwable as a String.)
- `String getAEFExceptionText(AEFException currentAEFException, int indentLevel)` (This method will return the information for this AEFException as a String.)
- `String toString()` (This method will print a summary of the exceptions.)
- `AEFException getOriginalException()` (Gets the original AEFException (last in the chain) on the list of chained exceptions.)
- `void setErrorObject(AEFError newErrorObject)` (Sets the AEF error object for this exception.)
- `AEFError getErrorObject()` (Gets the AEF error object for this exception.)

As noted above, `AEFException` has a `previousException` and a `nextException` attribute. This allows Store Integrator to chain these exceptions together in the case where more than one `AEFException` was generated. These exceptions are chained together with the oldest one at the end of the list. So the `AEFException` returned by any SI call is the exception that was most recently generated. If the intent of the handling code is just to log the exception, then the `AEFException` returned by the call should be logged. The error logging code will use the `toString` method of `AEFException` to correctly format all the exceptions on the list. If the code is trying to determine what the original problem was in order to take corrective action, then the original exception should be looked at. To get the original (oldest) exception you would have to call the `getOriginalException` method.

Once the original `AEFException` has been retrieved, the `getErrorCode`, `getErrorCodeExtended`, `getErrorKey`, and `getOriginalThrowable` methods should be called to try to determine how to handle the error. The `getErrorCode` method will always return a value from `AEFConst.java`. The `getErrorCodeExtended` method will return a value from `AEFConst.java` or zero if there is no extended code for this error. The `getErrorKey` method will return the key for this error if applicable. These keys are listed in the error bundle but for the most part they match the base application error codes (A003 or B007, etc.). If the return value from these methods is zero or null then more than likely a Java exception caused the error. The `getOriginalThrowable` method should be called to see what the cause of the error was.

Java Exceptions

As with any java program, Store Integrator programs need to be coded in such a way that exceptions are handled correctly. Aside from any error recovery code, it is also very important to log errors whenever an exception is detected. It is also important to note that unchecked exceptions do not require explicit try/catch blocks. So if all the error logging is done inside catch statements for checked exception, some very important messages could be missed. In order to avoid this, it is recommended that the `main()` method of your application has a `catch(Throwable)` block that will log any Exception that it catches.

Timeout Considerations

Some of the most difficult exceptions to recover from are timeout exceptions. These are `AEFException` objects with an error code of `AEFConst.OPERATION_TIMEOUT` or any Java exception that has the word `timeout` in its name. The most common is the `SocketTimeoutException` as a result of an RMI call. Here is a sample stack trace.

```
java.rmi.UnmarshalException: Error unmarshaling return header; nested exception is:
```

```

java.net.SocketTimeoutException: Read timed out
at sun.rmi.transport.StreamRemoteCall.executeCall(Unknown Source)
at sun.rmi.server.UnicastRef.invoke(Unknown Source)
at
com.ibm.retail.AEF.automation.SalesTransactionImpl_Stub.addTender(SalesTransactionImpl
_Stub.java:385)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
at java.lang.reflect.Method.invoke(Unknown Source)
at org.mozilla.javascript.NativeJavaMethod.call(Unknown Source)
at org.mozilla.javascript.ScriptRuntime.call(Unknown Source)
at org.mozilla.javascript.Interpreter.interpret(Unknown Source)
at org.mozilla.javascript.InterpretedScript.call(Unknown Source)
at org.mozilla.javascript.InterpretedScript.exec(Unknown Source)
at org.mozilla.javascript.Context.evaluateReader(Unknown Source)
at org.mozilla.javascript.Context.evaluateString(Unknown Source)
at
com.ibm.bsf.engines.javascript.JavaScriptEngine.eval(JavaScriptEngine.java:83)
at scripiter.AEFScript.processLine(AEFScript.java:765)
at scripiter.AEFScript.run(AEFScript.java:387)
Caused by: java.net.SocketTimeoutException: Read timed out
at java.net.SocketInputStream.socketRead0(Native Method)
at java.net.SocketInputStream.read(Unknown Source)
at java.io.BufferedInputStream.fill(Unknown Source)
at java.io.BufferedInputStream.read(Unknown Source)
at java.io.DataInputStream.readByte(Unknown Source)
... 17 more

```

The cause for most timeouts is either an enablement error or a slow system. In the case of an enablement error, a Store Integrator automation call is waiting for a condition that will never be satisfied. So after the appropriate timeout period, the exception is thrown. In order to correct this problem, either code or property files must be modified. Code changes are required if the condition list is incorrect. That is the automation call did not anticipate a possible outcome to the automation call. Property file changes are needed if a function code, state, or sub-state value is incorrect in a property file.

A special consideration must be given if the AEF error handling mode is set to "default". Since the application will wait for the operator, there is a chance that any one of the various calls made by the application will timeout. It is therefore recommended that if the default error handling mode is used that the value of any property in the config.properties file that ends with read.timeout be changed to 0. This will cause the read calls to wait indefinitely which will allow the operator time to take the necessary action. In the case of a slow system, the timeout values can be increased. The only danger is that if the cause of the timeout is really an enablement error, the operator will have to wait that much longer for the exception.

The reason why the timeout exceptions are so hard to deal with is that when the exception is thrown, the Store Integrator call and the base application disconnect. Basically the Store integrator code will throw the exception which will work its way up to the caller. The base application in the case of a slow system will eventually execute the automation call and get to the right state. So after the timeout, the Store Integrator application will have to try to figure out the outcome of the command that threw the timeout exception. How this is done will vary from one automation call to the next. But the basic idea would be to keep track of how many items, tenders, etc. had been added to the current transaction. Also state and sub-state information can be helpful. Monitoring the base application after the timeout error for any of these changes would give you an indication that the automation call had been processed.

Tender Recovery

When a client application tries to add a tender to a transaction, many things can go wrong. Such things as power failures, network problems, etc. can make it difficult to determine if the command succeeded or not. Also an exception or a timeout may be received because the system is behind on processing requests. Unfortunately, there is only so much that can be done programmatically to determine the real status of the tender request. Here are some suggestions of things to do to try to determine if the tender went through.

Before adding a tender, the client code should get information about the transaction that includes the transaction number, the number of tenders, transaction totals, all pertinent tender information. This data needs to be logged to persistent storage.

If a client receives either a `RemoteException` or an `AEFException` while trying to add a credit tender to the transaction, the tender operation is considered “in-flight” on the POS application. The client may be able to follow the following steps to determine whether the credit went through with approval or not.

- After the error, use the `POSAutomationProvider.getTransaction` method to see if there is a transaction in progress. If there is no transaction in progress, then the tender was taken and the transaction completed. (Tender satisfied the balance due). If there is a transaction in progress, then continue with the next step.
- If there is a transaction in progress, then call the `Transaction.getNumberOfTenders` and `Transaction.getTransactionTotals` methods and compare the results to the numbers from step one. If these numbers changed, then the tender was accepted. Otherwise continue with the next step.
- At this point, the status of the request to add the tender is not known. An error should be logged asking for somebody to use the tools provided by the base application to determine the status of the tender request.

Sample Code

The following code shows how to catch an `AEFException` (line 110), determine what the error codes are (lines 112-119), and recover from the error (lines 123-147). It also shows how to log the `Throwable` objects as described above to make sure that even unchecked exceptions are logged (lines 209-214).

```
1.  /*
2.   * AEFExample2
3.   *
4.   * 07/14/04
5.   *
6.   * Copyright:
7.   * Licensed Materials - Property of IBM
8.   * "Restricted Materials of IBM"
9.   * 5724-AEF
10.  * (C) Copyright IBM Corp. 2004.
11.  *
12.  */
13. package com.userco.retail.client.example2;
14.
15. import com.ibm.retail.si.util.*;
16. import com.ibm.retail.AEF.server.*;
17. import com.ibm.retail.AEF.session.*;
18. import com.ibm.retail.AEF.automation.*;
19. import com.ibm.retail.AEF.factory.*;
20.
21. import java.rmi.*;
22. import java.util.*;
23.
```

```

24. import org.apache.commons.logging.Log;
25. import org.apache.commons.logging.LogFactory;
26.
27. public class AEFExample2
28. {
29.
30.     /*
31.     * The main routine
32.     *
33.     * @param args an array of strings for command line parameters
34.     */
35.     public static void main(String[] args)
36.     {
37.         String errorText = null;
38.
39.         System.out.println("Starting AEFExample2...");
40.         try
41.         {
42.             SessionServer server = AEFBase.getInstance().getSessionServer();
43.             if (server != null)
44.             {
45.                 //connected
46.                 System.out.println("Obtained SessionServer");
47.                 AEFSession session = server.getAvailableSession();
48.                 if (session != null)
49.                 {
50.                     System.out.println("Obtained session. Terminal number is " +
51.                                     session.getTerminalNumber());
52.
53.                     System.out.println("Get an automation provider");
54.                     POSAutomationProvider automation =
55.                         session.getPOSAutomationProvider();
56.
57.                     // wait until the session is ready for commands
58.                     System.out.println("Wait until session is ready for
commands...");
59.                     boolean sessionReady = session.waitForReady(10000);
60.                     int retryCount = 0;
61.                     while (!sessionReady)
62.                     {
63.                         retryCount++;
64.                         System.out.println("Retrying... #" + retryCount);
65.                         sessionReady = session.waitForReady(10000);
66.                         if (retryCount > 5)
67.                         {
68.                             break;
69.                         }
70.                     }
71.                     if (sessionReady)
72.                     {
73.                         System.out.println("Session is ready for commands.");
74.
75.                         // initialize
76.                         System.out.println("Initialize the session to signon on
state.");
77.                         automation.initialize(POSAutomationProvider.SIGNON);
78.
79.                         // logon with default - set op/pw in userlogon.properties
on 4690
80.                         System.out.println("Logon using default logon.");
81.                         Operator op = automation.logon();
82.                         System.out.println("Logged on as " + op.getInfo().getID());
83.
84.                         // start a new transaction
85.                         System.out.println("Start a new transaction.");
86.                         SalesTransaction t1 = automation.startTransaction();
87.                         System.out.println("Transaction" +
t1.getTransactionInfo().getTransactionID() + "started.  ");
88.
89.                         // add item 33 to the transaction
90.
91.

```

```

92.         try
93.         {
94.             System.out.println("Add an item.");
95.             ArrayList itemList = t1.addItem("33");
96.             if ((itemList != null) && (!itemList.isEmpty()))
97.             {
98.                 Iterator it = itemList.iterator();
99.                 while (it.hasNext())
100.                 {
101.                     Item item = (Item)(it.next());
102.                     System.out.println("Item added.  Description
= " +
103. item.getInfo().getDescription());
104.                 }
105.
106.                 System.out.println("This is transaction " +
107. t1.getTransactionInfo().getTransactionID() + ".");
108.             }
109.         }
110.         catch (AEFException aie)
111.         {
112.             // Get the oldest exception in the list.
113.
114.             AEFException originalException =
aie.getOriginalException();
115.
116.             // Get the error codes.
117.
118.             int errorCode = originalException.getErrorCode();
119.             int extendedErrorCode =
originalException.getErrorCodeExtended();
120.
121.             // See if we can get around this error.
122.
123.             if (errorCode == AEFConst.INVALID_ARGUMENT &&
extendedErrorCode == AEFConst.ITEM_PRICE_REQUIRED)
124.             {
125.                 String tempPrice = getPrice();
126.
127.                 ItemIdentifier itemID =
(ItemIdentifier)(FactoryImpl.createObject(ItemIdentifier
.CLASS_KEY));
128.                 itemID.setItemCode("33");
129.                 itemID.setItemCodeType(AEFConst.VELOCITY);
130.                 itemID.setPrice(tempPrice);
131.
132.                 // Try to add the item again.
133.
134.                 System.out.println("Add an item (second try).");
135.                 ArrayList itemList = t1.addItem(itemID);
136.                 if ((itemList != null) && (!itemList.isEmpty()))
137.                 {
138.                     Iterator it = itemList.iterator();
139.                     while (it.hasNext())
140.                     {
141.                         Item item = (Item)(it.next());
142.                         System.out.println("Item added.
Description = " +
143. item.getInfo().getDescription());
144.                     }
145.
146.                     System.out.println("This is transaction " +
147. t1.getTransactionInfo().getTransactionID() + ".");
148.                 }
149.             }
150.         }
151.
152.         // void transaction

```

```

153.         System.out.println("Void transaction.");
154.         tl.voidTransaction();
155.
156.         // logoff this operator
157.         System.out.println("Logoff.");
158.         automation.logoff();
159.
160.         // release to make the session available again
161.         System.out.println("Release the session.");
162.         session.release();
163.         session=null;
164.     }
165.     else
166.     {
167.         errorText = "Timed out waiting for session ready";
168.         System.err.println(errorText);
169.         if (log.isWarnEnabled())
170.         {
171.             log.warn(errorText);
172.         }
173.     }
174. }
175. else
176. {
177.     errorText = "Unable to obtain AEFSession. ";
178.     System.err.println(errorText);
179.     if (log.isWarnEnabled())
180.     {
181.         log.warn(errorText);
182.     }
183. }
184. }
185. else
186. {
187.     // couldn't connect -- URL is incorrect or server is down
188.
189.     errorText = "Unable to connect to AEF SessionServer. ";
190.     System.err.println(errorText);
191.     if (log.isWarnEnabled())
192.     {
193.         log.warn(errorText);
194.     }
195. }
196. }
197. catch (AEFException ae)
198. {
199.     errorText = "An AEF exception occurred: ";
200.     System.err.println(errorText + ae.getMessage());
201.     log.error(errorText, ae);
202. }
203. catch (RemoteException re)
204. {
205.     errorText = "A remote exception occurred: ";
206.     System.err.println(errorText + re.getMessage());
207.     log.error(errorText, re);
208. }
209. catch (Throwable e)
210. {
211.     errorText = "A throwable object was caught: ";
212.     System.err.println(errorText + e.getMessage());
213.     log.error(errorText, e);
214. }
215. System.out.println("AEFExample2 complete.");
216. System.exit(0);
217. }
218.
219. /**
220.  * Gets the item's price.
221.  *
222.  * @return String The item's price.
223.  */
224. public static String getPrice()

```

```

225.      {
226.          // We will assume that there is a way to get the price from some other
source.
227.          // The point of this example is to show how to check and handle the
error.
228.          // Not how to get the price.
229.
230.          return "12.00";
231.      }
232.
233.      private static Log log = LogFactory.getLog(AEFExample2.class);
234.  }

```

Building, Debugging and Packaging

Task: Using Properties File Configuration

The behavior of AEF can be changed by modifying the values in various properties files. In this task you will learn how to change the behavior of all the terminals, or a subset of terminals. This subset will include real terminals, a specific terminal, or a group of terminals which belong to a session role.

Properties Files Bundles

The AEF bundles are sets of properties files that are used to configure AEF. Each bundle has a starting file that includes a link to the next file in the bundle (nextFile=). Each file is read in order. If the value of a property is specified in multiple files, then last value read will be used. This allows users to override AEF default values. AEF includes the following bundles:

- Automation (automation.properties, appautomation.properties, and userautomation.properties)
- Class (classes.properties, appclass.properties, and userclass.pro)
- Configuration (config.properties, appconfig.properties, and userconfig.properties)
- Error (error.properties, apperror.properties and usererror.properties)
- Function Code (fcode.properties, appfcode.properties, and userfcode.properties)
- Internalization (i18ntext.properties, appi18ntext.properties, and useri18ntext.properties)
- Key Sequence (sequence.properties, appseq.properties, and userseq.properties)
- Logon (applogon.properties, and userlogon.properties)
- Session (session.properties, appsession.properties, and usersession.properties)
- State (states.properties, appstate.properties, and userstates.properties)
- Substate (appsubstates.properties, and usersubstates.properties)
- Tender Map (tendermap.properties, aptendermap.properties, and usertendermap.properties)

All the AEF base properties files are included in the AEF.JAR file. The APP*.properties files are in the application specific Jar files (AEFGSA.JAR, AEFS.AJAR, or AEFACE.JAR). Any USER*.properties files must also be included in a Jar. It is recommended that user property overrides be put into one of the following jar files:

User Jar File	Effective For
user.jar	System wide configuration properties
usersim.jar	Property changes which effect virtual terminal sessions
usertof.jar	Property changes which effect real terminals sessions which are configured to use JavaTOF
userreal.jar	Property changes which effect real terminal sessions which are not configured to use JavaTOF

Table 14 User Jar File Names

The default AEF response files already include user.jar plus the correct user*.jar files in the classpath. Since these user jar files are not distributed as part of AEF, they must be created by the user.

For a complete list of the properties in each file see “Appendix B. AEF Property Files” on page 158 in this guide, or use the commands described below to extract the files from the Jar files.

Manipulating Jar Files

A Jar file is a Zip compatible file that contains Java class files and other resources (like images and properties files). Since these are Zip files, any zip file utility can be used to manipulate these files. If you do not have access to any Zip utilities, the JDK comes with a “jar” command.

The following table lists some commands for working with Jar files.

Function	Command
Create a new jar file containing user properties.	<code>d:/si/userjar>jar cvf user.jar *</code>
To list the contents of the aef.jar file.	<code>d:/si/si_jars>jar tvf aef.jar</code>

Table 15 Jar Commands

For more information about the Jar command, see the documentation that came with your JDK or type `jar` on the command line to get a list of all the command line options.

Changing the Default User ID and Password for All Terminals

The Automation Provider includes a logon ID that uses default operator ID and passwords. When the `POSAutomationProvider.logon` method is called without arguments, the terminal number is used to lookup an operator id and password. The Logon bundle defines the default operator ID and password for each terminal number, and is made-up of the `applogon.properties`, and `userlogon.properties` files. The `applogon.properties` file that ships in the SI CDs contain the following lines:

```
#termno=id,password
001=1,1
002=1,1
100=1,1
999=2,2
nextFile=userlogon
```

The first line (`#termno=id,password`) is a comment line. Any line that starts with a # is considered to be a comment. Lines 2 through 4 specify that terminals 1, 2, and 100 should use operator ID 1 and a password of 1 if the logon method is called without any arguments. Line 5 specifies that terminal 999 should use ID 2, and a password of 2 if the logon method is called without any arguments. The last line specifies the next file in the bundle.

In order to change these values, you need to create a file called `c:/si/userjar/userlogon.properties`. Since this is typically the last file in the chain, it does not need a `nextFile` line. In this example you will use the terminal number as the ID and password. So the file would look like this.

```
# Terminal 1 user 1 password 1
001=1,1

# Terminal 2 user 2 password 2
002=2,2
```

```
# Terminal 3 user 3 password 3
003=3,3
```

```
# Terminal 4 user 4 password 4
004=4,4
```

Now that you have a userlogon.properties file, you need to include it in a jar file that is in the classpath. Since you want this change to affect all the terminals, you will add this file to the user.jar file. To create a new user.jar file that includes the userlogon.properties file, switch into the c:/si/userjar directory, and use the command

```
d:/si/userjar>jar cvf user.jar *
```

If you have already created a user.jar file, use the command

```
d:/si/userjar>jar uvf user.jar userlogon.properties
```

The resulting user.jar file now needs to be copied to the c:\user directory on the 4690 controller (including setting file distribution attributes), and the terminals will need to be reloaded. In the case of TOF terminals, the appropriate TOF batch file must be run before the terminals are reloaded.

Changing the Error Handling Mode for Some Terminals

In this task you will set the error handling mode for all real terminals to “default”. This means that the system will not try to automatically correct or clear any automation errors. Instead the system will wait for the operator to take the correct action. The error handling mode is set in the automation bundle. The default appautomation.properties file looks like this.

```
#
# This file lists all the valid configuration properties for the
# POSAutomationProvider.
#
# For each property, the default value is provided. A list of valid property values
# may be
# specified by including the list inside [] characters. When a list is provided, the
# first
# value in the list becomes the default property value.
#
com\\ibm\\retail\\AEF\\automation\\error.handling.mode=[automatic,default,callback]
com\\ibm\\retail\\AEF\\automation\\error.max.errors.to.handle=10
com\\ibm\\retail\\AEF\\automation\\item.age.restriction=[true,false]
nextFile=userautomation
```

So in order to override the value for com\\ibm\\retail\\AEF\\automation\\error.handling.mode you must create a file called c:/si/userjar/userautomation.properties. This file will contain the following line:

```
com\\ibm\\retail\\AEF\\automation\\error.handling.mode=[default,automatic,callback]
```

Now that you have a userautomation.properties file, you need to include it in a jar file that is in the classpath. Since you want this change to affect only the real terminals, you will add this file to the usertof.jar, and userreal.jar files. Since the CSS terminals do not include either of these two Jar files in their classpath, they will not be affected by this change. To create a new usertof.jar file that includes the userautomation.properties file, use the command

```
d:/si/userjar>jar cvf usertof.jar userautomation.properties
```

To create a new userreal.jar file that includes the userautomation.properties file, use the command

```
d:/si/userjar>jar cvf userreal.jar userautomation.properties
```

To add the userautomation.properties file to an existing usertof.jar file, use the command

```
d:/si/userjar>jar uvf usertof.jar userautomation.properties
```

To add the userautomation.properties file to an existing userreal.jar file, use the command

```
d:/si/userjar>jar uvf userreal.jar userautomation.properties
```

The resulting Jar files now need to be copied to the c:\user directory on the 4690 controller (including setting file distribution attributes), and the terminals will need to be reloaded. In the case of TOF terminals, the appropriate TOF batch file must be run before the terminals are reloaded.

Changing the Session Trace Level of a Specific Terminal

Both virtual and real sessions write trace information into a “session trace” buffer that may be used for troubleshooting. In this task you will set the session trace level for a specific terminal session. This could be helpful if you are experiencing a problem that requires more log information to troubleshoot than what is normally generated by the system. You could set all the terminals to do full logging but this could hurt performance. By setting the level of one terminal, you get the additional data and do not degrade the performance of the other terminals.

The session trace level is controlled by the line

```
trace.level=COARSE
```

in the session.properties file. So in order to override this value for only one terminal (terminal 100 for example) you need to create a file called c:/si/userjar/usersession_100.properties. This file will only contain the line

```
trace.level=FINE
```

Once you have created this file, you need to get it into a Jar file that is in the classpath. To create a new user.jar file that includes the usersession_100.properties file, use the command

```
d:/si/userjar>jar cvf user.jar *
```

To add the usersession_100.properties file to an existing user.jar file, use the command

```
d:/si/userjar>jar uvf user.jar usersession_100.properties
```

The resulting user.jar file now needs to be copied to the c:\user directory on the 4690 controller (including setting file distribution attributes), and the terminal will need to be reloaded. If terminal 100 uses TOF, then the appropriate TOF batch file must be run before the terminal is reloaded.

Changing the Session Trace Buffer Size for Terminals by Role

In this task you will set the session trace buffer size for a group of terminals that share the same session role. A role is a logical group of terminal sessions. In this example we will create a role for the all the self checkout terminal sessions. The session trace buffer size is controlled by the line

```
trace.buffer.size=1000
```

in the session.properties file. So in order to override this value for only the self checkout terminals (terminals 200, 201, and 300 to 310 for example) you need to create two new files. One is c:/si/userjar/usersessionrole.properties. This file is used to define the role and its terminals. The file will contain the line

```
SelfCheckout=200,201,300-310
```

The second file is called c:/si/userjar/usersession_SelfCheckout.properties. This file will only contain the line

```
trace.buffer.size=5000
```

Once we have created these two files, we need to get them into a Jar file that is in the classpath. To create a new user.jar file that includes the usersessionrole.properties , and usersession_SelfCheckout.properties files, use the command

```
d:/si/userjar>jar cvf user.jar *
```

To add the usersessionrole.properties, and usersession_SelfCheckout.properties files to an existing user.jar file, use the command

```
d:/si/userjar>jar uvf user.jar usersessionrole.properties  
usersession_SelfCheckout.properties
```

The resulting user.jar file now needs to be copied to the c:/user directory on the 4690 controller (including setting file distribution attributes), and the terminals will need to be reloaded. If these terminals use TOF, then the appropriate TOF batch file must be run before the terminal is reloaded.

Task: Packaging Your Code

In order for user written code to be run, it must be included in one of the files in the classpath for the application. The default response files shipped on the SI CD include several Jar files in the 4690 OS Java classpath that are intended for user code. These Jar files include c:/user/user.jar for system wide properties, and C:/user/usersim.jar, c:/user/usertof.jar, or c:/user/userreal.jar for terminal type specific classes. By including your class and properties files in these Jar files, the response files will not have to be modified. The following example will show you how to add some classes to the user.jar file.

Adding Classes to a Jar File

For the purpose of building AEF user extensions in Java, all user Java class code should be in a directory hierarchy rooted at c:/si/userjar. For example, a user extension class called com.userco.retail.AEF.action.UserLogonActionImpl.java should be placed in c:/si/user/com/userco/retail/AEF/action/UserLogonActionImpl.java. When it is compiled, the -d option should be used to place the .class file in c:/si/userjar/com/userco/retail/AEF/action/UserLogonActionImpl.class.

To create a new user.jar file that includes all the user property files and class files in the “userjar” directory hierarchy, use the command

```
d:/si/userjar>jar cfv user.jar *
```

Task: Using the 4690 Debugger for POS Applications under CSS

This task describes the process for debugging a 4690 POS application such as SA or GSA while running under CSS. The steps required are:

- Ensure the proper level of the IBM 4690 Application Debugger is installed on the 4690 controller. The Debugger must be Version 4 Level 18 or higher.
- Perform the necessary AEF configuration.
- Execute the debugger

Installing the 4690 Debugger

The IBM 4690 Application Debugger must be installed on the 4690 controller. The Debugger must be Version 4 Level 18 or higher. See the Debugger installation diskette for installation instructions.

Configuring the AEF

There are two configuration properties which must be set to debug a virtual session. In the config.properties bundle, the “debug.css.pos.appl” must be set to true. In the session.properties bundle, the “pos.sales.application” property must be set to execute “EWTTerm.286”.

First, create (or edit) the file

```
d:/si/userjar/userconfig.properties
```

and add the line

```
debug.css.pos.appl=true
```

Choose the virtual session number to be debugged. Assuming the session number is 201, create (or edit the file)

```
d:/si/userjar/usersession_201.properties
```

and add the line

```
pos.sales.application=c:/debug/ewtterm.286
```

Once you have created or edited the properties files, you need to get them into a Jar file that is on the classpath for CSS. To create a new user.jar file that includes the files, use the command

```
d:/si/userjar>jar cvf user.jar *
```

To add the files to an existing user.jar file, use the command

```
d:/si/userjar>jar uvf user.jar usersession_201.properties  
userconfig.properties
```

On the 4690, stop CSS if it is running, and copy the user.jar file to the c:\user directory on the 4690 controller (including setting file distribution attributes), then restart CSS.

Running the Debugger

To execute the debugger on the 4690 controller, execute the line:

```
c:/debug>debug /Vnnn xxxxxxxx.286
```

where **nnn** is the session number to be debugged (201 in this example), and **xxxxxxx** is the terminal sales application name.

Chapter 4. AEF Enabling a POS Application

The following steps must be taken in order for a POS application to run in the SIF environment:

1. Modify AEF configuration settings. It may be necessary to provide AEF configuration information, for example, default operator id and passwords for terminal sessions.
2. Install POS application hooks. The POS application must include hooks to provide application XML data as appropriate. Events are expected for start and end of transactions, items added to the transaction, transaction totals updates, etc.
3. Enable user extensions and additional features. The AEF must be programmed to perform POSAutomationProvider operations such as “logon”, “startTransaction”, “addItem”, “addTender”, etc. This involves providing “action” classes which understand the POS application. Each action class determines if the POS application is in the proper state to perform the action, sends the appropriate key sequences to the application, and defines the expected results. If the POS application includes user extensions or additional features, then the action classes must be extended or replaced so that the action is performed properly in the context of the user extensions.

AEF Configuration Settings

The AEF is configured via property name/value pairs which reside in various property files. See “Appendix B. AEF Property Files” on page 158 for information describing each of the configuration properties.

Some of these configuration properties give the AEF information about how to interact with the underlying POS application. The following sections describe these properties.

Default Operator ID/Passwords

One of the functions of the POSAutomationProvider is to allow a client program to logon to a terminal session. The client may specify an operator ID and password. If no operator ID and password is specified, the AEF will use the default operator ID and password which is configured for that session’s terminal number. For details on defining default operator ID and passwords, see “Changing the Default User ID and Password for All Terminals” on page 134.

Automatic Manager Override Number

When a POSAutomationProvider method is called to perform some POS activity, the underlying POS application may generate an error or condition that can be overridden via a manager’s override. The POSAutomationProvider is configured by default to automatically perform the manager’s override. See “Automation Bundle” on page 158 for information on the configuration property which controls this behavior.

When the POS application requires the keying of a manager override number, the AEF must be configured with a valid manager override number. See “Automation Bundle” on page 158 for information on how to configure a default manager override number.

Tender Mappings

The AEF must be able to make correlations between tender descriptions (which is a string), and tender type and tender varieties (which are numeric values). For example, the IBM 4680-4690 Supermarket Application may have a tender definition for “Credit” with tender type equal to 4, and tender variety equal to 1.

For each tender definition in the application, a tender mapping needs to be configured in the AEF. However, for this release of SI, only credit tenders are supported, so tender mappings for other tender types are not required.

See “Tender Map Bundle” on page 278 for information on how to configure application tender mappings.

POS Application Hooks

The AEF receives information from the POS application in the form of XML events. Events are sent when an operator logs on or off, when a transaction starts or ends, when items are added or removed from the transaction, etc.

The following IBM POS applications and features have been AEF enabled, and include the hooks necessary to send the application data to the AEF:

Application/Feature	Base Level	Efix
IBM 4680-4690 General Sales Application (GSA) <ul style="list-style-type: none">• 4610 Printer Feature• GSA TOF	M001	4197
IBM 4680-4690 Supermarket Application (SA) <ul style="list-style-type: none">• 4610 Printer Feature• SA TOF• EM PRPQ• Value Pack 2001• EFTE Feature	N001 <ul style="list-style-type: none">• N001• N001• M001• L001• J001	4171 <ul style="list-style-type: none">• 4181• 4171• 4173• 4174• 4172
IBM SurePOS Application Client/Server Environment for 4690 OS (ACE)	V3	

Table 16 AEF Enabled Applications

For GSA and SA, the required terminal sales event hooks are included, but must be linked into the terminal sales program. Instructions for linking the AEF event hooks are included under “*Selecting terminal sales components*” in *Store Integrator User’s Guide*. For ACE, the required terminal sales event hooks are included in a special version of the terminal sales executable called `JSIFTS10.386` which is included with ACE V3.

POS Application Events

The POS applications generate the following events which are transmitted to the AEF JVM. These XML events are used to generate Java events (see “

Event and Listener Types” on page 47) which are fired by the POSDataProvider. See “Appendix D. POS Application XML Events” on page 316 for a list of POS application XML events.

The figure below illustrates a typical AEF enabled IBM POS application.

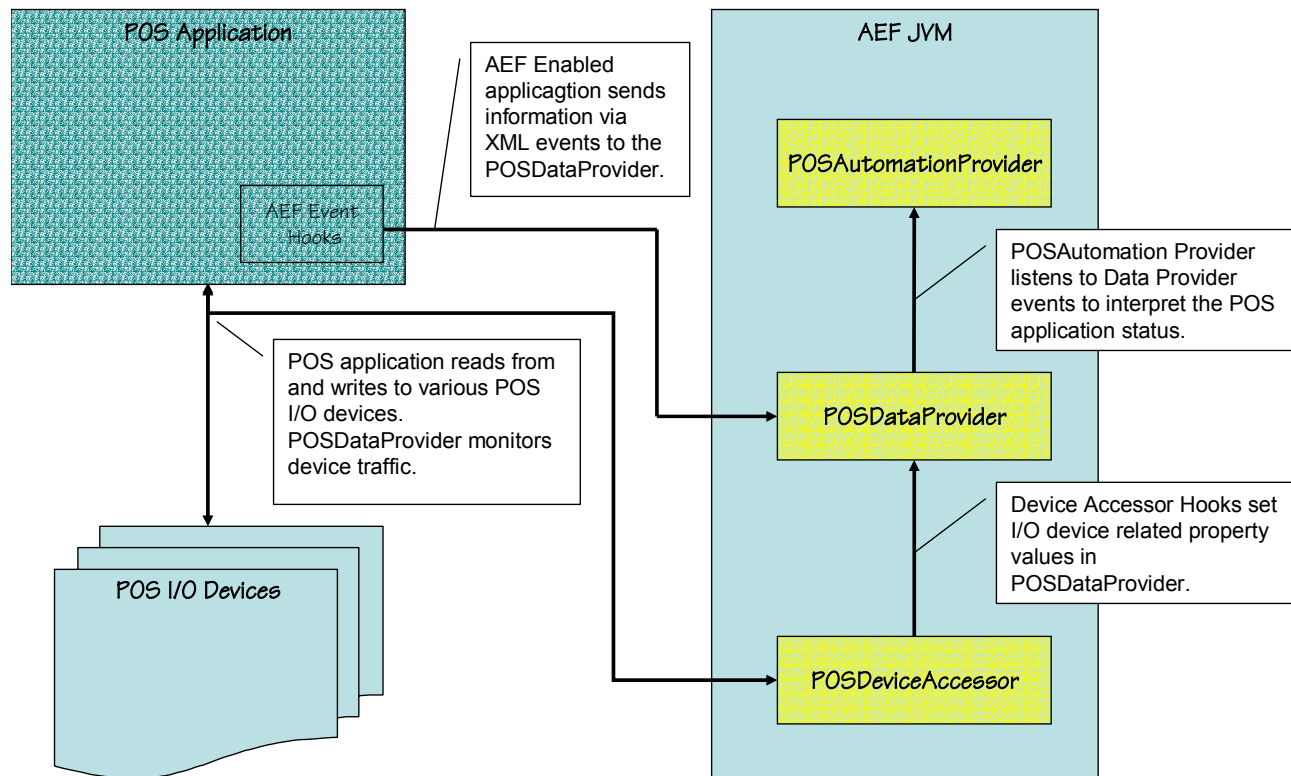


Figure 23 AEF Enabled Application

The AEF has two sources of information for POS application data.

1. The POS application is enhanced with "AEF event hooks" which transmit XML events from the CBASIC or C application to the AEF JVM. For example, events are transmitted when transactions are initiated or terminated, when items are added or removed from a transaction, when transaction totals are updated, when receipt print lines are generated, when the scale weight changes, when the application substate changes, etc.
2. When POS I/O devices are redirected, the AEF may install Device Accessor hooks to monitor device data and status. For example, the AEF monitors the contents to the line displays, the IOP (4690 Input/Output Processor) state, POSPrinter status, the last labels scanned, cash drawer status, the track data for the last card swiped, etc.

Enabling User Extensions and Additional Features

The process of “AEF Enabling” user extensions and additional application features involves two main aspects:

1. Modifying the POS application to send any additional required data to the AEF via existing or new XML messages.
2. Override or extend the AEF Automation Provider API as required to be able to deal with the POS application modifications.

Sending Additional POS Data to the AEF

An analysis of the AEF Enablement requirements will determine the additional POS application data that must be provided to the AEF via XML messages. It may be possible to extend existing XML messages as described in “Task: Adding Additional Data to an Event” on page 74. If this is not possible, then new XML messages may be sent to the AEF from the POS application as described in “Task: Sending a New XML Event” on page 81.

Overriding or Extending the Automation Provider API

As previously explained in this guide, the Automation Provider API works by injecting input to the POS application, and monitoring properties such as IO state, application substate, operator display contents, etc. to watch for conditions indicating success, failure, or errors which must be handled. Action classes are defined which check preconditions, inject key sequences, and wait for the results, or intermediate conditions which must be handled.

Each POS application must be examined in regards to the AEF to determine if the following might impact the AEF.

1. New operator prompts.
2. Key sequence changes.

Either of these might be the result of user extensions to the POS application, or applications features which are not AEF enabled.

New Operator Prompts

These will have to be analyzed on a case by case basis. If the new prompt has the potential to occur during the executing of an Automation Provider API call, it will probably have to be dealt with. Dealing with the new prompt will depend on the nature of the prompt.

Typically, the action class will need to be overridden to deal with the prompt. In some cases, it may be possible to add error helper configuration for the prompt without overriding the action class.

The process of overriding an action will typically involve modifying a sequence’s definition of good and bad expected conditions. The condition indicating the new prompt may be considered a good condition if it will be handled within the action. Alternatively, the condition indicating the new prompt may be defined as a bad condition if it will be dealt with via an error helper.

See “Task: Extending Existing Automation Provider APIs” on page 101 for an example of overriding an action class.

Key Sequence Changes

If a user modification to the POS application changed a key sequence, it will be necessary to reflect that change in the AEF. In the simplest case, the key sequence definition in the AEF can be modified in the sequence.properties bundle.

If the key sequence is a new key sequence, the same considerations for replacing the action implementation apply as discussed in the previous section.

If the key sequence has been modified to accept additional data, it may be necessary to modify the key sequence definition in the sequence.properties bundle, as well as replace the action implementation to send the additional data required by the modified key sequence. “Task: Extending Existing Automation Provider APIs” on page 101 is an example of overriding an action to handle a modified key sequence which requires additional data.

Appendix A. Error Codes

AEFConst Error Codes

The following table lists AEFException error codes. AEFException error codes are sometimes paired with extended error codes. The constants are defined in com.ibm.retail.si.util.AEFConst.

Primary Error Code	Constant	Possible Extended Error Codes	Notes
-102	APPLICATION_NOT_IN_PROPER_STATE		The requested operation is not valid given the current state of the POS application.
-103	COUPON_EXPIRED		
-104	NO_SUCH_PROPERTY		An attempt was made to set a property value on a property name that does not exist.
-105	LOYALTY_CARD_EXPIRED		
-106	OPERATION_TIMEOUT		The AEF did not detect that the operation completed within its configured timeout period. The operation may still complete, even though the timeout error has been thrown back to the client.
-107	NETWORK_FAILURE		
-109	UNSUPPORTED_TRANSACTION_TYPE		For this release, only regular sales transactions are supported.
-111	INVALID_ARGUMENT	WEIGHT_AND_QUANTITY_MUTUALLY_EXCLUSIVE PRICE_AND_DEAL_PRICE_MUTUALLY_EXCLUSIVE DEAL_PRICE_AND_DEAL_QUANTITY_REQUIRED INVALID_DEAL_QUANTITY INVALID_QUANTITY INVALID_PRICE INVALID_DEAL_PRICE INVALID_WEIGHT INVALID_EXPIRY_DATE INVALID_TENDER_AMOUNT INVALID_CURRENCY_AMOUNT ARGUMENT_REQUIRED ACCOUNT_NUMBER_REQUIRED EXPIRATION_DATE_REQUIRED CREDIT_CARD_TYPE_REQUIRED INVALID_LOYALTY_NUMBER ITEM_CLASS_REQUIRED ITEM_CLASS_PROHIBITED ITEM_STOCK_REQUIRED ITEM_STOCK_PROHIBITED ITEM_DEPARTMENT_REQUIRED ITEM_DEPARTMENT_PROHIBITED ITEM_PRICE_REQUIRED ITEM_WEIGHT_REQUIRED ITEM_QUANTITY_REQUIRED ITEM_WEIGHT_PROHIBITED	The extended error code indicates which argument was in error.

Primary Error Code	Constant	Possible Extended Error Codes	Notes
		DATA_OUT_OF_RANGE INVALID_AMOUNT EXTENDED_PRICE_TOO_LARGE INVALID_DEPARTMENT INVALID_CHARGE_PLAN AMOUNT_REQUIRED INVALID_FEE_AMOUNT ITEM_CODE_REQUIRED INVALID_TAX_CODE TAX_CODE_REQUIRED ID_REQUIRED INVALID_TRANSACTION_TYPE INVALID_ACCOUNT_NUMBER INVALID_PASSWORD PASSWORD_REQUIRED TRANSACTION_TYPE_REQUIRED INVALID_AUTHORIZATION_CODE INVALID_DEPARTMENT_TOTALS_LIST_NUMBER INVALID_BATCH_NUMBER INVALID_TERMINAL_NUMBER INVALID_TENDER_TYPE ITEM_PRICE_OR_QUANTITY_REQUIRED INVALID_ITEM_CODE INVALID_TARE INVALID_DISCOUNT_CODE INVALID_ID AUTHORIZATION_CODE_REQUIRED INVALID_VOUCHER_NUMBER VOUCHER_NUMBER_REQUIRED INVALID_REASON INVALID_DATE VOLUME_PROHIBITED VOLUME_REQUIRED INVALID_VOLUME EXTENDED_PRICE_PROHIBITED TRACK_DATA_REQUIRED TRACK_DATA_PROHIBITED ID_PROHIBITED INVALID_ITEM_IDENTIFIER_TYPE ITEM_QUANTITY_PROHIBITED REASON_CODE_PROHIBITED INVALID_INITIALIZE_STATE NEW_PASSWORD_PROHIBITED REASON_CODE_REQUIRED INVALID_CARD_ID CARD_ID_REQUIRED VALUE_EXCEEDS_CONNECTION_TIMEOUT	
-112	SIGNATURE_REQUIRED		
-113	MANAGER_OVERRIDE_REQUIRED		
-116	ITEM_NOT_FOUND		
-118	FILE_NOT_FOUND		
-122	INVALID_ACTION_REQUEST		
-123	INPUT_NOT_ALLOWED		The current state of the POS application does not allow input, therefore the requested operation could not be completed.
-124	SYSTEM_BUSY		
-125	MISSING_CONFIG_VALUE		The exception text indicates which configuration property is missing.
-127	WAIT_INTERRUPTED		

Primary Error Code	Constant	Possible Extended Error Codes	Notes
-129	ANOTHER_OPERATOR_SIGNED_ON		The logon request could not be completed because another operator is already logged onto the session.
-130	AGE_RESTRICTED_ITEM		
-131	ITEM_IS_TIME_RESTRICTED		
-134	PROPERTY_NOT_SUPPORTED		
-135	INVALID_PROPERTY_VALUE		
-137	INVALID_LISTENER_TYPE		
-138	NO_LISTENER_SUPPORT		
-139	EVENT_LISTENER_MISMATCH		
-141	UNEXPECTED_STATE		
-142	AUTHORIZATION_REQUIRED		
-152	TENDER_NOT_ACCEPTED	TENDER_NOT_AUTHORIZED TOO_LONG_IN_STAND_IN CARD_EXPIRED UNKNOWN_SERVICER SERVICER_CLOSED CARD_TYPE_UNKNOWN RISK_1 RISK_2 RISK_3 RISK_4 VERIFICATION_TIMEOUT CREDIT_NOT_AVAILABLE INVALID_CARD_DATA CALL_FOR_AUTHORIZATION PAYMENT_SYSTEM_OFFLINE CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD CUSTOMER_CANCELLED_TENDER_AT_PINPAD MSR_TRACK_DATA_REQUIRED MSR_TRACK_DATA_NOT_ALLOWED SCANNED_DATA_NOT_ALLOWED CUSTOMER_ALTERED_AMOUNT_AT_PINPAD CANCELLED_BY_OPERATOR LOYALTY_NUMBER_REQUIRED PIN_PAD_REQUIRED TENDER_EXPIRED VOUCHER_NOT_YET_VALID PIN_PAD_PROHIBITED	
-153	FILE_ACCESS_FAILED		The POS application failed to access a required file.
-157	TONE_ERROR		
-158	MSR_ERROR		
-159	APPLICATION_LIMIT_EXCEEDED	MAX_NUMBER_OF_ITEMS MAX_NUMBER_OF_PRICE_CHANGES TRANSACTION_TOTAL_TOO_LARGE DISCOUNT_PERCENTAGE_TOO_SMALL DISCOUNT_PERCENTAGE_TOO_LARGE DEPARTMENT_RETURN_LIMIT CHANGE_AMOUNT_LIMIT TRANSACTION_LIMIT ITEM_LIMIT DISCOUNT_LIMIT ACCOUNT_TENDER_LIMIT TENDER_AMOUNT_LIMIT NUMBER_OF_TENDERS_LIMIT STAND_IN_COUNT_LIMIT	

Primary Error Code	Constant	Possible Extended Error Codes	Notes
		STAND_IN_AMOUNT_LIMIT TENDER_FLOOR_LIMIT LOYALTY_POINTS_LIMIT NUMBER_OF_COUPONS_LIMIT SUSPENDED_TRANSACTION_LIMIT TARE_WEIGHT_TOO_LARGE NEGATIVE_TRANSACTION_BALANCE FOODSTAMP_TENDER_IN_EXCESS_OF_ FOODSTAMP_BALANCE DAILY_LOYALTY_CARD_USAGE_LIMIT	
-160	ERROR_HANDLER_OTHER_ERROR		
-161	ITEM_NOT_RETURNABLE		
-162	NO_ITEM_MATCH_FOR_COUPON		The coupon could not be redeemed because the POS application requires a matching item, but there was not matching item.
-163	COUPON_VALUE_EXCEEDS_ITEM_VALUE		
-164	RETURN_COUPON_BEFORE_ITEM_VOID		Could not void the item because a coupon associated with the item must be returned first.
-165	VOID_MUST_MATCH_PREVIOUS		Voided item must match an item previously in the transaction.
-166	SAME_ERROR_TWICE		The error handler processed an error, but the POS application responded by returning to the same error. This error is thrown to avoid a loop.
-167	TOO_MANY_ERRORS		The error handler exceeded the configured number of sequential errors to try to handle.
-168	ERROR_HANDLER_EXCEPTION_DETECTED		
-169	ERROR_HANDLER_FATAL_ERROR		
-170	ERROR_HANDLER_OVERRIDE_ERROR		
-171	INVALID_MANAGER_OVERRIDE_NUMBER		
-172	ITEM_NOT_FOR_SALE		
-173	DISPLAY_ERROR		
-174	PASSWORD_EXPIRED		
-175	CUSTOMER_MUST_SIGN_DOCUMENT		
-176	UNCHECKED_EXCEPTION		The AEF threw an unchecked exception which was not expected.
-177	ERROR_ACCESSING_FACTORY		
-179	MICR_ERROR	ACCOUNT_NUMBER_ERROR	
-180	SERVER_NO_SESSION_AVAILABLE		The session server could not locate a CSS instance with available sessions.
-181	FACTORY_NO_SESSIONS_AVAILABLE		
-182	FACTORY_SESSION_NOT_READY		
-183	UNSUPPORTED_ERROR_HANDLING_MODE		
-184	POS_APP_FAILURE	FILE_IO_ERROR INVALID_APPLICATION_DATA UNRECOGNIZED_PRINT_CHARACTERS OUT_OF_MEMORY PAYMENT_SYSTEM_ERROR HARDWARE_PROBLEM	The POS application experienced an unexpected error.
-185	CONFIG_ERROR	KEYSEQUENCE_NOT_FOUND INVALID_KEYSEQUENCE_EXPRESSION	Invalid AEF configuration caused an error. See the extended error

Primary Error Code	Constant	Possible Extended Error Codes	Notes
		FCODE_NOT_FOUND INVALID_KEY_SEQUENCE NONNUMERIC_FUNCTION_CODE UNABLE_TO_LOAD_PROPERTIES_FILE NO_ERROR_HELPER_CONFIGURED TAX_TABLE_NOT_FOUND MISSING_DESCRIPTOR TAX_OPTION_MISMATCH MISSING_EXCHANGE_RATE UNKNOWN_MICR_FORMAT UNDEFINED_TENDER_VARIETY NO_DEFAULT_ID_CONFIGURED FACTORY_ERROR NO_DEFAULT_MANAGER_OVERRIDE_PASSWORD	code to determine which configuration setting is in error.
-187	ENABLEMENT_ERROR	AUTHORIZATION_NUMBER_MISMATCH	
-190	KEYLOCK_ERROR	MANAGER_KEY_REQUIRED MANAGER_KEY_NOT_REMOVED	
-191	PRINTER_ERROR	JOURNAL_ERROR BUFFER_FULL COVER_OPEN PAPER_LOW	
-196	SCALE_ERROR	ITEM_IS_ON_THE_SCALE PRICE_PER_POUND_REQUIRED PRICE_PER_KILOGRAM_REQUIRED REPLACE_ITEM WEIGH_ITEM CHECK_ITEM_PRICE ITEM_CANNOT_BE_WEIGHED CHECK_WEIGHT_PERSONALIZATION CHECK_OVERRIDE_PRICE ITEM_NOT_ON_SCALE_CORRECTLY CHECK_CONFIGURATION PREVIOUS_ITEM_ON_SCALE REWEIGH_ITEM DEVICE_OFFLINE	
-208	UNSUPPORTED_OPERATION		The requested operation is not implemented in this release.
-209	NO_SESSION_FACTORIES_FOUND		A request for a session could not be honored because the session server has not established contact with any session factories.
-213	COIN_DISPENSER_ERROR		
-214	CASH_DRAWER_ERROR	CLOSE_DRAWER NONE_AVAILABLE	
-215	KEYBOARD_ERROR		
-217	CLIENT_SESSION_MISMATCH		Client attempted to release a session, but the session had already been released.
-234	PIN_PAD_ERROR	COMMUNICATION_ERROR PIN_COULD_NOT_BE_OBTAINED DEVICE_IS_LOADING LOAD_ERROR UNSUPPORTED_CAPABILITY	
-236	USER_DEFINED		
-251	RMI_REGISTRY_FAILURE		
-252	RMI_NAMING_FAILURE		
-253	DEVICE_HANDLER_FAILURE		
-254	DEVICE_HOOK_ERROR		
-255	DEVICE_MANAGER_ERROR		

Primary Error Code	Constant	Possible Extended Error Codes	Notes
-256	VERIFY_SIGNATURE		
-259	JAVA_POS_EXCEPTION		An error was reported by the JavaPOS device layer. See the target or cause within the AEFException to get the details of the JposException.
-260	INVALID_TIMEOUT_VALUE		
-261	JAVA_INVOCATION_FAILED		
-263	REMOTE_SESSION_SERVER_ERROR		
-264	INPUT_CANCELLED		
-301	INVALID_DISPATCH_QUEUE		
-302	API_VALID_FOR_TSS_ONLY		The API may only be called when running CSS.
-303	ERROR_STARTING_POS_APPLICATION		
-304	ERROR_STOPPING_POS_APPLICATION		
-305	SESSION_NOT_ACTIVE		
-306	ERROR_CREATING_SESSION		
-307	ERROR_CREATING_DEVICE_SERVER		
-308	MSR_HOOK_SWIPE_ERROR		
-309	ERROR_CREATING_GENERAL_AGENT		
-310	PROCEDURE_NOT_ALLOWED	TENDER_TYPE_NOT_ALLOWED PAYMENT_ALREADY_ENTERED NOT_ALLOWED_OFFLINE NOT_ALLOWED_REENTRY CHARGE_PLAN_NOT_ALLOWED NOT_AUTHORIZED OPERATOR_ALREADY_ACTIVE NOT_ALLOWED_TRAINING ACCOUNT_PAID_IN_FULL EXTERNAL_TENDER_AUTHORIZATION_S USPENDED ACCOUNT_NOT_YET_AVAILABLE FULLSCREEN_NOT_SUPPORTED PINPAD_NOT_AVAILABLE CLOSING_ACCOUNTING_PERIOD PICKUP_NEEDED TILL_EXCHANGE_NEEDED MONITOR_ALREADY_ACTIVE HOME_STORE_MUST_REDEEM_LOYALT Y_POINTS MORE_LOYALTY_POINTS_NEEDED LOYALTY_NUMBER_REQUIRED MINIMUM_SALE_NOT_SATISFIED RETRIEVE_TRANSACTION_WHERE_SUS PENDED UNABLE_TO_RETRIEVE_TRANSACTION TRANSACTION_ALREADY_RETRIEVED SAME_OPERATOR_MUST_RETRIEVE_TR ANSACTION NON_WIC_ITEM NON_WIC_TENDER TAX_EXEMPTION_NOT_ALLOWED_FOR_I TEM DISCOUNT_NOT_ALLOWED_FOR_ITEM ITEM_DISCOUNTS_NOT_ALLOWED MICR_DATA_NOT_ALLOWED EBT_NOT_ALLOWED HOST_REQUEST_PENDING VALUE_CARDS_MUST_BE_VOIDED NOT_APPROVED_BY_PAYMENT_SYSTEM CASHBACK_ON_FINAL_TENDER_ONLY	The procedure is not currently valid for the reason identified by the extended error code.

Primary Error Code	Constant	Possible Extended Error Codes	Notes
		PAYMENT_SYSTEM_OFFLINE LOYALTY_COUPON_NOT_APPLICABLE_T O_CUSTOMER_STATUS_LEVEL TRANSACTION_NOT_ACTIVE TRANSACTION_CANNOT_BE_VOIDED NO_SUCH_SUSPENDED_TRANSACTION TRANSACTION_IN_PROGRESS TRANSACTION_ALREADY_IN_PROGRESS NOT_IMPLEMENTED	
-311	JPOS_NOEXIST		
-312	TERMINAL_DISABLED		
-313	DEVICE_NOT_REDIRECTED		The requested Workstation procedure requires the relevant device to be configured as a redirected device.
-314	ERROR_ACCESSING_GENERAL_AGENT		
-320	TIMEOUT_WAITING_FOR_AVAILABLE_SESSION		
-321	SESSION_NOT_AVAILABLE		
-322	SESSION_READY_WAIT_TIMEOUT		The session did not become ready within the configured timeout period.

Table 17 AEFException Error Codes

AEFConst Extended Error Codes

The following table lists extended error codes used by the `AEFException` class.

Extended Error Code	Constant	Notes
0	NONE	
-1003	RMI_NAMING_REGISTRY_RETURNED_NULL	
-1004	RMI_REMOTE_EXCEPTION	
-1005	RMI_URL_EXCEPTION	
-1006	RMI_BIND_EXCEPTION	
-1007	REMOTE_EXCEPTION	
-1008	SESSION_FACTORY_REMOTE_EXCEPTION	
-1009	APPLICATION_NOT_IN_PROPER_SUBSTATE	
-1010	NO_SUCH_LOGGER	
-1011	INVALID_LOGGER_LEVEL	
-1012	CLASS_NOT_FOUND	
-1013	APPLICATION_ALREADY_ACTIVE	
-1014	APPLICATION_NOT_ACTIVE	
-1015	TSS_ERROR	
-1016	MSR_SET_TO_DECODE	The AEF does not support configuring the JavaPOS MSR control to decode the fields.
-1017	MSR_NOT_ENABLED	Can't inject MSR track data into the MSR device hook because the MSR is not enabled.
-1250	WEIGHT_AND_QUANTITY_MUTUALLY_EXCLUSIVE	
-1251	PRICE_AND_DEAL_PRICE_MUTUALLY_EXCLUSIVE	
-1252	DEAL_PRICE_AND_DEAL_QUANTITY_REQUIRED	
-1253	INVALID_DEAL_QUANTITY	
-1254	INVALID_QUANTITY	
-1255	INVALID_PRICE	
-1256	INVALID_DEAL_PRICE	
-1257	INVALID_WEIGHT	
-1258	INVALID_EXPIRY_DATE	
-1259	INVALID_TENDER_AMOUNT	
-1260	INVALID_CURRENCY_AMOUNT	
-1261	ARGUMENT_REQUIRED	
-1262	ACCOUNT_NUMBER_REQUIRED	
-1263	EXPIRATION_DATE_REQUIRED	
-1264	CREDIT_CARD_TYPE_REQUIRED	
-1265	INVALID_LOYALTY_NUMBER	
-1267	ITEM_CLASS_REQUIRED	
-1268	ITEM_CLASS_PROHIBITED	
-1269	ITEM_STOCK_REQUIRED	
-1270	ITEM_STOCK_PROHIBITED	
-1271	ITEM_DEPARTMENT_REQUIRED	
-1272	ITEM_DEPARTMENT_PROHIBITED	
-1273	ITEM_PRICE_REQUIRED	
-1274	ITEM_PRICE_PROHIBITED	
-1275	ITEM_QUANTITY_REQUIRED	
-1276	ITEM_QUANTITY_PROHIBITED	
-1277	DATA_OUT_OF_RANGE	

Extended Error Code	Constant	Notes
-1278	INVALID_AMOUNT	
-1279	EXTENDED_PRICE_TOO_LARGE	
-1280	INVALID_DEPARTMENT	
-1281	INVALID_CHARGE_PLAN	
-1282	AMOUNT_REQUIRED	
-1283	INVALID_FEE_AMOUNT	
-1284	ITEM_CODE_REQUIRED	
-1285	INVALID_TAX_CODE	
-1286	TAX_CODE_REQUIRED	
-1287	ID_REQUIRED	
-1288	INVALID_TRANSACTION_TYPE	
-1289	INVALID_ACCOUNT_NUMBER	
-1290	INVALID_PASSWORD	
-1291	PASSWORD_REQUIRED	
-1292	TRANSACTION_TYPE_REQUIRED	
-1293	INVALID_AUTHORIZATION_CODE	
-1294	INVALID_DEPARTMENT_TOTALS_LIST_NUMBER	
-1295	INVALID_BATCH_NUMBER	
-1296	INVALID_TERMINAL_NUMBER	
-1297	INVALID_TENDER_TYPE	
-1298	ITEM_PRICE_OR_QUANTITY_REQUIRED	
-1299	INVALID_ITEM_CODE	
-1300	INVALID_TARE	
-1301	INVALID_DISCOUNT_CODE	
-1302	INVALID_ID	
-1303	AUTHORIZATION_CODE_REQUIRED	
-1304	INVALID_VOUCHER_NUMBER	
-1305	VOUCHER_NUMBER_REQUIRED	
-1306	INVALID_REASON	
-1307	INVALID_DATE	
-1308	VOLUME_PROHIBITED	
-1309	VOLUME_REQUIRED	
-1310	EXTENDED_PRICE_REQUIRED	
-1311	EXTENDED_PRICE_PROHIBITED	
-1312	TRACK_DATA_REQUIRED	
-1313	TRACK_DATA_PROHIBITED	
-1314	ID_PROHIBITED	
-1315	INVALID_ITEM_IDENTIFIER_TYPE	
-1316	ITEM_QUANTITY_PROHIBITED	
-1317	REASON_CODE_PROHIBITED	
-1318	INVALID_INITIALIZE_STATE	
-1319	NEW_PASSWORD_PROHIBITED	
-1320	REASON_CODE_REQUIRED	
-1321	INVALID_CARD_ID	
-1322	CARD_ID_REQUIRED	
-1323	VALUE_EXCEEDS_CONNECTION_TIMEOUT	
-1500	TENDER_NOT_AUTHORIZED	
-1503	TOO_LONG_IN_STAND_IN	
-1508	CARD_EXPIRED	
-1509	UNKNOWN_SERVICER	

Extended Error Code	Constant	Notes
-1510	SERVICER_CLOSED	
-1511	CARD_TYPE_UNKNOWN	
-1515	RISK_1	
-1516	RISK_2	
-1517	RISK_3	
-1518	RISK_4	
-1519	VERIFICATION_TIMEOUT	
-1520	CREDIT_NOT_AVAILABLE	
-1522	INVALID_CARD_DATA	
-1523	CALL_FOR_AUTHORIZATION	
-1524	PAYMENT_SYSTEM_OFFLINE	
-1525	CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD	
-1526	CUSTOMER_CANCELLED_TENDER_AT_PINPAD	
-1527	MSR_TRACK_DATA_REQUIRED	
-1528	MSR_TRACK_DATA_NOT_ALLOWED	
-1530	SCANNED_DATA_NOT_ALLOWED	
-1531	CUSTOMER_ALTERED_AMOUNT_AT_PINPAD	
-1532	CANCELLED_BY_OPERATOR	
-1533	LOYALTY_NUMBER_REQUIRED	
-1534	PIN_PAD_REQUIRED	
-1535	TENDER_EXPIRED	
-1536	VOUCHER_NOT_YET_VALID	
-1537	PIN_PAD_PROHIBITED	
-1650	JOURNAL_ERROR	
-1651	BUFFER_FULL	
-1652	COVER_OPEN	
-1653	PAPER_LOW	
-1750	FILE_IO_ERROR	
-1751	INVALID_APPLICATION_DATA	
-1752	UNRECOGNIZED_PRINT_CHARACTERS	
-1753	OUT_OF_MEMORY	
-1754	PAYMENT_SYSTEM_ERROR	
-1755	HARDWARE_PROBLEM	
-1800	MANAGER_KEY_REQUIRED	
-1801	MANAGER_KEY_NOT_REMOVED	
-1820	ITEM_IS_ON_THE_SCALE	
-1821	PRICE_PER_POUND_REQUIRED	
-1822	PRICE_PER_KILOGRAM_REQUIRED	
-1824	REPLACE_ITEM	
-1825	WEIGH_ITEM	
-1826	CHECK_ITEM_PRICE	
-1827	ITEM_CANNOT_BE_WEIGHED	
-1828	CHECK_WEIGHT_PERSONALIZATION	
-1829	CHECK_OVERRIDE_PRICE	
-1830	ITEM_NOT_ON_SCALE_CORRECTLY	
-1831	CHECK_CONFIGURATION	
-1832	PREVIOUS_ITEM_ON_SCALE	
-1833	REWEIGH_ITEM	
-1834	DEVICE_OFFLINE	
-1875	KEYSEQUENCE_NOT_FOUND	

Extended Error Code	Constant	Notes
-1876	INVALID_KEYSEQUENCE_EXPRESSION	
-1877	FCODE_NOT_FOUND	
-1878	INVALID_KEY_SEQUENCE	
-1879	NONNUMERIC_FUNCTION_CODE	
-1880	UNABLE_TO_LOAD_PROPERTIES_FILE	
-1881	NO_ERROR_HELPER_CONFIGURED	
-1882	TAX_TABLE_NOT_FOUND	
-1883	MISSING_DESCRIPTOR	
-1884	TAX_OPTION_MISMATCH	
-1885	MISSING_EXCHANGE_RATE	
-1886	UNKNOWN_MICR_FORMAT	
-1887	UNDEFINED_TENDER_VARIETY	
-1888	NO_DEFAULT_ID_CONFIGURED	
-1889	FACTORY_ERROR	
-1890	NO_DEFAULT_MANAGER_OVERRIDE_PASSWORD	
-2000	MAX_NUMBER_OF_ITEMS	
-2001	MAX_NUMBER_OF_PRICE_CHANGES	
-2002	TRANSACTION_TOTAL_TOO_LARGE	
-2003	DISCOUNT_PERCENTAGE_TOO_SMALL	
-2004	DISCOUNT_PERCENTAGE_TOO_LARGE	
-2005	DEPARTMENT_RETURN_LIMIT	
-2006	CHANGE_AMOUNT_LIMIT	
-2007	TRANSACTION_LIMIT	
-2008	ITEM_LIMIT	
-2009	DISCOUNT_LIMIT	
-2010	ACCOUNT_TENDER_LIMIT	
-2011	TENDER_AMOUNT_LIMIT	
-2012	NUMBER_OF_TENDERS_LIMIT	
-2013	STAND_IN_COUNT_LIMIT	
-2014	STAND_IN_AMOUNT_LIMIT	
-2015	TENDER_FLOOR_LIMIT	
-2016	LOYALTY_POINTS_LIMIT	
-2017	NUMBER_OF_COUPONS_LIMIT	
-2018	SUSPENDED_TRANSACTION_LIMIT	
-2019	TARE_WEIGHT_TOO_LARGE	
-2020	NEGATIVE_TRANSACTION_BALANCE	
-2021	FOODSTAMP_TENDER_IN_EXCESS_OF_FOODSTAMP_BALANCE	
-2022	DAILY_LOYALTY_CARD_USAGE_LIMIT	
-2200	TENDER_TYPE_NOT_ALLOWED	
-2201	PAYMENT_ALREADY_ENTERED	
-2202	NOT_ALLOWED_OFFLINE	
-2203	NOT_ALLOWED_REENTRY	
-2204	CHARGE_PLAN_NOT_ALLOWED	
-2205	NOT_AUTHORIZED	
-2206	OPERATOR_ALREADY_ACTIVE	
-2207	NOT_ALLOWED_TRAINING	
-2208	ACCOUNT_PAID_IN_FULL	
-2209	EXTERNAL_TENDER_AUTHORIZATION_SUSPENDED	
-2210	ACCOUNT_NOT_YET_AVAILABLE	
-2211	FULLSCREEN_NOT_SUPPORTED	

Extended Error Code	Constant	Notes
-2212	PINPAD_NOT_AVAILABLE	
-2213	CLOSING_ACCOUNTING_PERIOD	
-2214	PICKUP_NEEDED	
-2215	TILL_EXCHANGE_NEEDED	
-2216	MONITOR_ALREADY_ACTIVE	
-2217	HOME_STORE_MUST_REDEEM_LOYALTY_POINTS	
-2218	MORE_LOYALTY_POINTS_NEEDED	
-2219	MINIMUM_SALE_NOT_SATISFIED	
-2220	RETRIEVE_TRANSACTION_WHERE_SUSPENDED	
-2221	UNABLE_TO_RETRIEVE_TRANSACTION	
-2222	TRANSACTION_ALREADY_RETRIEVED	
-2223	SAME_OPERATOR_MUST_RETRIEVE_TRANSACTION	
-2224	NON_WIC_ITEM	
-2225	NON_WIC_TENDER	
-2226	TAX_EXEMPTION_NOT_ALLOWED_FOR_ITEM	
-2227	DISCOUNT_NOT_ALLOWED_FOR_ITEM	
-2228	ITEM_DISCOUNTS_NOT_ALLOWED	
-2229	MICR_DATA_NOT_ALLOWED	
-2230	EBT_NOT_ALLOWED	
-2231	HOST_REQUEST_PENDING	
-2232	VALUE_CARDS_MUST_BE_VOIDED	
-2233	NOT_APPROVED_BY_PAYMENT_SYSTEM	
-2234	CASHBACK_ON_FINAL_TENDER_ONLY	
-2235	LOYALTY_COUPON_NOT_APPLICABLE_TO_CUSTOMER_STATUS_LEVEL	
-2237	TRANSACTION_NOT_ACTIVE	
-2238	TRANSACTION_CANNOT_BE_VOIDED	
-2239	NO_SUCH_SUSPENDED_TRANSACTION	
-2240	TRANSACTION_IN_PROGRESS	
-2241	TRANSACTION_ALREADY_IN_PROGRESS	
-2242	NOT_IMPLEMENTED	
-2400	COMMUNICATION_ERROR	
-2401	PIN_COULD_NOT_BE_OBTAINED	
-2402	DEVICE_IS_LOADING	
-2403	LOAD_ERROR	
-2404	UNSUPPORTED_CAPABILITY	
-2450	ACCOUNT_NUMBER_ERROR	
-2475	AUTHORIZATION_NUMBER_MISMATCH	
-2650	CLOSE_DRAWER	
-2651	NONE_AVAILABLE	

Table 18 AEFException Extended Error Codes

Appendix B. AEF Property Files

The AEF bundles are sets of property files that are used to configure AEF. Each bundle has a starting file that includes a link to the next file in the bundle (`nextFile=`). Each file is read in order. If the value of a property is specified in multiple files, the last value read will be used. This allows users to override AEF default values.

The property file chains start with a base property file (for example, `config.properties`). The base property file uses a `nextFile=appxxxx` to chain into an application specific property file (for example, `appconfig.properties`). The purpose of the application specific property files are to provide a place for configuration values that are unique depending upon the underlying POS application. Each application property file chains into a user property file using the syntax `nextFile="userxxxx"` (for example, `userconfig.properties`). The user property files allow any base or application settings to be overridden.

4690 Filename Restriction

The filenames on the C: drive of a 4690 system are restricted to an 8.3 format. This makes it hard to give the property files meaningful names. To overcome this, the files in the bundles are included in Jar files. This way the OS deals with 8.3 filenames while inside those files, Java can use long filenames. All the base property files are in the AEF.JAR file. The APP*.properties files are in the application specific Jar files (AEFGSA.JAR, AEFSJA.JAR, or AEFACE.JAR). Any USER*.properties files must also be included in a Jar file. It is recommended that these property files be put into files called C:/user/user.jar for system wide properties, and C:/user/usersim.jar (for properties effecting only virtual sessions), C:/user/usertof.jar (for properties effecting real terminals which are configured to use JavaTOF), or C:/user/userreal.jar (for real terminals not using JavaTOF). The default AEF response files already include C:/user/user.jar plus the correct C:/user/user*.jar files in the classpath. Since the user .jar files are not distributed as part of AEF, they must be created by the user.

Automation Bundle

The Automation bundle is used to specify how automation calls should behave when errors are encountered or when an override is needed. The files that makeup the Automation bundle are `automation.properties`, `appautomation.properties`, and `userautomation.properties`.

For each property, the default value is provided. A list of valid property values may also be specified by including the list inside [] characters. When a list is provided, the first value in the list becomes the default property value.

Automation.properties

The `automation.properties` file includes the following properties.

- **`com\ibm\retail\AEF\automation\automatic.manager.override=[true,false]`** – This property is used to tell AEF whether to perform a manager's override automatically or not.

`nextFile=appautomation` – This property is used to tell AEF the name of the next property file in the bundle chain.

Appautomation.properties (ACE)

The `appautomation.properties` file includes the following properties.

- **com\ibm\retail\AEF\automation\automatic.manager.override.number=9** – This is the manager override number to be used with automatic error handling.
- **com\ibm\retail\AEF\automation\automatic.operator.override=[true,false]** – In automatic error handling, true indicates an operator override will be done when needed.
- **com\ibm\retail\AEF\automation\item.age.restriction=[true,false]** – This property tells AFE how to handle age restricted items. If set to true, AEF will bypass the age restrict code. This could be used on a shopping cart device where we would not want the shopper to be the one to verify their age.
- **com\ibm\retail\AEF\automation\item.time.restriction=[true,false]** – Not Currently Used
- **com\ibm\retail\AEF\automation\automatic.special.signon=[true,false]** – When true, a transaction start or an item entry in special sign off, will do auto sign on.
- **com\ibm\retail\AEF\automation\error.handling.mode=[automatic,default,call back]** – This property tells AEF what error handling mode to use when handling errors. See the Error Handler section of this document for more information.
- **com\ibm\retail\AEF\automation\error.max.errors.to.handle=10** – This property tells AFE how many errors in a row to try to handle before it quits and throws an exception.

nextFile=userautomation – This property is used to tell AEF the name of the next property file in the bundle chain.

Appautomation.properties (GSA)

The appautomation.properties file includes the following properties.

- **com\ibm\retail\AEF\automation\error.handling.mode=[automatic,default,call back]** – This property tells AEF what error handling mode to use when handling errors. See the Error Handler section of this document for more information.
- **com\ibm\retail\AEF\automation\error.max.errors.to.handle=10** – This property tells AFE how many errors in a row to try to handle before it quits and throws an exception.
- **com\ibm\retail\AEF\automation\item.age.restriction=[true,false]** – This property tells AFE how to handle age restricted items. If set to true, AEF will bypass the age restrict code. This could be used on a shopping cart device where we would not want the shopper to be the one to verify their age.

nextFile=userautomation – This property is used to tell AEF the name of the next property file in the bundle chain.

Appautomation.properties (SA)

The appautomation.properties file includes the following properties.

- **com\ibm\retail\AEF\automation\automatic.manager.override.number=1** – This is the manager override number to be used with automatic error handling
- **com\ibm\retail\AEF\automation\automatic.operator.override=[true,false]** – In automatic error handling, true indicates an operator override will be done when needed.
- **com\ibm\retail\AEF\automation\item.age.restriction=[true,false]** – This property tells AFE how to handle age restricted items. If set to true, AEF will bypass the age restrict code. This could be used on a shopping cart device where we would not want the shopper to be the one to verify their age.
- **com\ibm\retail\AEF\automation\item.time.restriction=[true,false]** – Not Currently Used

- **com\ibm\retail\AEF\automation\automatic.special.signon=[true,false]** – When true, a transaction start or an item entry in special sign off, will do auto sign on.
- **com\ibm\retail\AEF\automation\error.handling.mode=[automatic,default,call back]** – This property tells AEF what error handling mode to use when handling errors. See the Error Handler section of this document for more information.
- **com\ibm\retail\AEF\automation\error.max.errors.to.handle=10** – This property tells AEF how many errors in a row to try to handle before it quits and throws an exception.

nextFile=userautomation – This property is used to tell AEF the name of the next property file in the bundle chain.

Class Bundle

The Class bundle is used to tell AEF what Java class should be used to do a specific job. The files that makeup the Class bundle are classes.properties, appclass.properties, and userclass.pro.

For each property, a key is provided followed by an equal sign and a class name. These class names will be used to create objects as needed. By changing the values for these keys, the classes of the objects created by AEF can be replaced with user classes.

Classes.properties

The classes.properties file includes the following properties.

General Classes

- AEFSession=com.ibm.retail.AEF.session.AEFSessionImpl
- AEFSessionFactory=com.ibm.retail.AEF.factory.AEFSessionFactoryImpl
- AEFSessionFactoryInfo=com.ibm.retail.AEF.factory.AEFSessionFactoryInfo
- AEFSessionPool=com.ibm.retail.AEF.factory.AEFSessionPool
- ActionFactory=com.ibm.retail.AEF.factory.ActionFactory
- ActionProcessor=com.ibm.retail.AEF.automation.ActionProcessorImpl
- ApplicationDataConnector=com.ibm.retail.AEF.data.ApplicationDataConnectorImpl
- CouponInfo=com.ibm.retail.AEF.automation.CouponInfoImpl
- CouponLineItem=com.ibm.retail.AEF.automation.CouponImpl
- CreditIdentifier=com.ibm.retail.AEF.automation.CreditIdentifierImpl
- CreditInfo=com.ibm.retail.AEF.automation.CreditInfoImpl
- Customer=com.ibm.retail.AEF.automation.CustomerImpl
- CustomerDetector=com.ibm.retail.AEF.automation.CustomerDetectorImpl
- CustomerInfo=com.ibm.retail.AEF.automation.CustomerInfoImpl
- DefaultErrorHandler=com.ibm.retail.AEF.util.DefaultErrorHandler
- DefaultTender=com.ibm.retail.AEF.automation.TenderImpl
- Discount=com.ibm.retail.AEF.automation.DiscountImpl
- DiscountInfo=com.ibm.retail.AEF.automation.DiscountInfoImpl
- DiskInfoLogger=com.ibm.retail.AEF.util.DiskInfoLoggerImpl
- HardTotalsHook=com.ibm.retail.AEF.device.HardTotalsHookImpl
- Item=com.ibm.retail.AEF.automation.ItemImpl
- ItemCodeParser=com.ibm.retail.AEF.automation.ItemCodeParserImpl
- ItemIdentifier=com.ibm.retail.AEF.automation.ItemIdentifierImpl
- ItemInfo=com.ibm.retail.AEF.automation.ItemInfoImpl
- KeyboardMapper=com.ibm.retail.AEF.workstation.OS4690KeyboardMapperImpl
- KeySequenceDriver=com.ibm.retail.AEF.workstation.JIOPKeySequenceDriverImpl
- LineItemDetector=com.ibm.retail.AEF.automation.LineItemDetectorImpl
- LineItemInfo=com.ibm.retail.AEF.automation.LineItemInfoImpl

- LoadBalancer=com.ibm.retail.AEF.server.DefaultLoadBalancerImpl
- LoyaltyIdentifier=com.ibm.retail.AEF.automation.LoyaltyIdentifierImpl
- Operator=com.ibm.retail.AEF.automation.OperatorImpl
- OperatorAuthorization=com.ibm.retail.AEF.automation.OperatorAuthorizationImpl
- OperatorDetector=com.ibm.retail.AEF.automation.OperatorDetectorImpl
- OperatorIdentifier=com.ibm.retail.AEF.automation.OperatorIdentifierImpl
- OperatorInfo=com.ibm.retail.AEF.automation.OperatorInfoImpl
- Points=com.ibm.retail.AEF.automation.PointsImpl
- PointsInfo=com.ibm.retail.AEF.automation.PointsInfoImpl
- POSAutomationProvider=com.ibm.retail.AEF.automation.POSAutomationProviderImpl
- POSDataProvider=com.ibm.retail.AEF.data.POSDataProviderImpl
- restrict=com.ibm.retail.AEF.event.TimeIntervalImpl
- SetPropertyAction=com.ibm.retail.AEF.action.SetPropertyActionImpl
- SalesTransactionInfo=com.ibm.retail.AEF.automation.SalesTransactionInfoImpl
- StoreInfo=com.ibm.retail.AEF.automation.StoreInfoImpl
- SuspendedTransactionInfo=com.ibm.retail.AEF.automation.SuspendedTransactionInfoImpl
- TenderInfo=com.ibm.retail.AEF.automation.TenderInfoImpl
- TenderMapper=com.ibm.retail.AEF.automation.TenderMapperImpl
- SalesTransaction=com.ibm.retail.AEF.automation.SalesTransactionImpl
- SalesTransactionTotals=com.ibm.retail.AEF.automation.TransactionTotalsImpl
- SAXEventParser=com.ibm.retail.AEF.xml.SAXEventParser
- SessionServer=com.ibm.retail.AEF.server.SessionServerImpl
- SessionTrace=com.ibm.retail.AEF.util.SessionTraceImpl
- SimpleKeySequenceAction=com.ibm.retail.AEF.action.SimpleKeySequenceActionImpl
- SwipeMSRAction=[com.ibm.retail.AEF.action.SwipeMSRActionImpl](#)
- TenderDetector=com.ibm.retail.AEF.automation.TenderDetectorImpl
- TenderMapperImpl=com.ibm.retail.AEF.automation.TenderMapperImpl
- TransactionDetector=com.ibm.retail.AEF.automation.TransactionDetectorImpl
- Transaction=com.ibm.retail.AEF.automation.TransactionImpl
- TransactionIdentifier=com.ibm.retail.AEF.automation.TransactionIdentifierImpl
- TransactionInfo=com.ibm.retail.AEF.automation.TransactionInfoImpl
- TransactionTotalsSetter=com.ibm.retail.AEF.event.TransactionTotalsSetter
- InitializeAction=[com.ibm.retail.AEF.action.InitializeActionImpl](#)
- Workstation=com.ibm.retail.AEF.workstation.WorkstationImpl

Listener support classes

- GenericEventListenerSupport=com.ibm.retail.AEF.event.GenericEventListenerSupport
- CashReceiptListenerSupport=com.ibm.retail.AEF.event.CashReceiptListenerSupport
- OperatorListenerSupport=com.ibm.retail.AEF.event.OperatorListenerSupport
- CustomerListenerSupport=com.ibm.retail.AEF.event.CustomerListenerSupport
- PointsListenerSupport=com.ibm.retail.AEF.event.PointsListenerSupport
- CouponListenerSupport=com.ibm.retail.AEF.event.CouponListenerSupport
- DiscountListenerSupport=com.ibm.retail.AEF.event.DiscountListenerSupport
- StateChangeListenerSupport=com.ibm.retail.AEF.event.StateChangeListenerSupport
- TransactionTotalsListenerSupport=com.ibm.retail.AEF.event.TransactionTotalsListenerSupport
- TransactionStatusListenerSupport=com.ibm.retail.AEF.event.TransactionStatusListenerSupport
- ItemSalesListenerSupport=com.ibm.retail.AEF.event.ItemSalesListenerSupport
- TenderListenerSupport=com.ibm.retail.AEF.event.TenderListenerSupport

- ReportListenerSupport=com.ibm.retail.AEF.event.ReportListenerSupport
- ScaleListenerSupport=com.ibm.retail.AEF.event.ScaleListenerSupport
- OptionsListenerSupport=com.ibm.retail.AEF.event.OptionsListenerSupport
- WorkstationStatusListenerSupport=com.ibm.retail.AEF.event.WorkstationStatusListenerSupport
-

ApplicationDataConnector event classes (format: "xml-element"="event class name")

- alternateTaxCode=com.ibm.retail.AEF.event.TaxCodeImpl
- cashReceipt=com.ibm.retail.AEF.event.CashReceiptEventImpl
- coupon=com.ibm.retail.AEF.event.CouponEventImpl
- Coupon=com.ibm.retail.AEF.event.CouponEventImpl
- customer=com.ibm.retail.AEF.event.CustomerEventImpl
- dataEvent=com.ibm.retail.AEF.event.POSAppEventImpl
- departmentDefinition=com.ibm.retail.AEF.event.DepartmentDefinitionImpl
- genericEvent=com.ibm.retail.AEF.event.POSAppEventImpl
- itemDiscountReason=com.ibm.retail.AEF.event.DiscountReasonCodeImpl
- itemEvent=com.ibm.retail.AEF.event.ItemSalesEventImpl
- line=com.ibm.retail.AEF.event.ReportLineImpl
- lineItemDiscount=com.ibm.retail.AEF.event.DiscountEventImpl
- lock=com.ibm.retail.AEF.event.OperatorEventImpl
- loyaltyUpdate=com.ibm.retail.AEF.event.CustomerEventImpl
- manualTaxCode=com.ibm.retail.AEF.event.TaxCodeImpl
- noTaxCode=com.ibm.retail.AEF.event.TaxCodeImpl
- options=com.ibm.retail.AEF.event.OptionsEventImpl
- points=com.ibm.retail.AEF.event.PointsEventImpl
- priceOverrideReason=com.ibm.retail.AEF.event.ReasonCodeImpl
- refundReason=com.ibm.retail.AEF.event.ReasonCodeImpl
- report=com.ibm.retail.AEF.event.ReportEventImpl
- section=com.ibm.retail.AEF.event.ReportSectionImpl
- scale=com.ibm.retail.AEF.event.ScaleEventImpl
- signOff=com.ibm.retail.AEF.event.OperatorEventImpl
- signOn=com.ibm.retail.AEF.event.OperatorEventImpl
- state=com.ibm.retail.AEF.event.StateChangeEventImpl
- store=com.ibm.retail.AEF.event.StoreDefinitionImpl
- storeOptions=com.ibm.retail.AEF.event.StoreOptionsImpl
- tareCode=com.ibm.retail.AEF.event.TareCodeImpl
- tender=com.ibm.retail.AEF.event.TenderEventImpl
- tenderDefinition=com.ibm.retail.AEF.event.TenderDefinitionImpl
- terminalOptions=com.ibm.retail.AEF.event.TerminalOptionsImpl
- transactionDefinition=com.ibm.retail.AEF.event.TransactionDefinitionImpl
- transactionDiscount=com.ibm.retail.AEF.event.DiscountEventImpl
- transactionDiscountReason=com.ibm.retail.AEF.event.TransactionDiscountReasonCodeImpl
- transactionEnd=com.ibm.retail.AEF.event.TransactionStatusEventImpl
- transactionStart=com.ibm.retail.AEF.event.TransactionStatusEventImpl
- transactionSuspended=com.ibm.retail.AEF.event.TransactionStatusEventImpl
- transactionTaxChange=com.ibm.retail.AEF.event.TransactionStatusEventImpl
- transactionTotals=com.ibm.retail.AEF.event.TransactionTotalsEventImpl
- transactionUpdate=com.ibm.retail.AEF.event.TransactionStatusEventImpl
- transactionVoid=com.ibm.retail.AEF.event.TransactionStatusEventImpl
- unitOfWork=com.ibm.retail.AEF.event.POSAppEventImpl
- unlock=com.ibm.retail.AEF.event.OperatorEventImpl
- vatTaxCode=com.ibm.retail.AEF.event.TaxCodeImpl
- voidReason=com.ibm.retail.AEF.event.ReasonCodeImpl

- workstationStatus=com.ibm.retail.AEF.event.WorkstationStatusEventImpl

Tender Line Item Implementation classes

- AmericanExpressTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- AmexTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- CashTender=com.ibm.retail.AEF.automation.TenderImpl
- CheckTender=com.ibm.retail.AEF.automation.TenderImpl
- CreditTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- CreditCardTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- CreditPlanATender=com.ibm.retail.AEF.automation.TenderImpl
- CreditPlanBTender=com.ibm.retail.AEF.automation.TenderImpl
- CreditPlanCTender=com.ibm.retail.AEF.automation.TenderImpl
- CreditPlanDTender=com.ibm.retail.AEF.automation.TenderImpl
- DinersClubTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- DiscoverTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- DueBillTender=com.ibm.retail.AEF.automation.TenderImpl
- FoodStampsTender=com.ibm.retail.AEF.automation.TenderImpl
- GiftCertificateTender=com.ibm.retail.AEF.automation.TenderImpl
- MasterCardTender=com.ibm.retail.AEF.automation.CreditTenderImpl
- MiscellaneousTender=com.ibm.retail.AEF.automation.TenderImpl
- StoreCouponTender=com.ibm.retail.AEF.automation.TenderImpl
- Type16Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type17Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type18Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type19Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type20Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type21Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type22Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type23Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type24Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type25Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type26Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type27Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type28Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type29Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type30Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type31Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type32Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type33Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type34Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type35Tender=com.ibm.retail.AEF.automation.TenderImpl
- Type36Tender=com.ibm.retail.AEF.automation.TenderImpl
- TravelersCheckTender=com.ibm.retail.AEF.automation.TenderImpl
- VenderCouponTender=com.ibm.retail.AEF.automation.TenderImpl
- VisaTender=com.ibm.retail.AEF.automation.CreditTenderImpl

nextFile=appclass – This property is used to tell AEF the name of the next property file in the bundle chain.

Appclass.properties (ACE)

The appclass.properties file includes the following properties.

General Classes

- AEFErrorImpl=com.ibm.retail.AEF.util.ACEAEFErrorImpl
- AgeRestrictedHelper=com.ibm.retail.AEF.util.ACEAgeRestrictedHelper
- AppDefaultErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperDefault
- AppInit=com.ibm.retail.AEF.util.ACEAppInit
- ApplyDelayedCouponsAction=[com.ibm.retail.AEF.action.ACEApplyDelayedCouponsActionImpl](#)
- AuthorizationCodeErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperAuthorizationCode
- BadConditionsImpl=[com.ibm.retail.AEF.util.ACEBadConditionsImpl](#)
- CancelPreviousEntryAction=[com.ibm.retail.AEF.action.ACECancelPreviousEntryActionImpl](#)
- CardValidationNumberErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperCardValidationNumber
- CustomerEntryAction=[com.ibm.retail.AEF.action.ACECustomerEntryActionImpl](#)
- DescriptorsImpl=com.ibm.retail.AEF.util.ACEDescriptorsImpl
- ExpirationDateErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperExpirationDate
- FatalClearErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperFatalClear
- FatalErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperFatal
- ForcedLogoffAction=[com.ibm.retail.AEF.action.ACEForcedLogoffActionImpl](#)
- HardTotalsHook=com.ibm.retail.AEF.device.ACEHardTotalsHookImpl
- IgnoreErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperIgnore
- InformationalErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperInformational
- ItemEntryAction=[com.ibm.retail.AEF.action.ACEItemEntryActionImpl](#)
- ItemPromptErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperItemPrompt
- ItemReturnAction=[com.ibm.retail.AEF.action.ACEItemReturnActionImpl](#)
- ItemVoidAction=[com.ibm.retail.AEF.action.ACEItemVoidActionImpl](#)
- LevelOverrideErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperLevelOverride
- LogoffAction=[com.ibm.retail.AEF.action.ACELogoffActionImpl](#)
- LogonAction=[com.ibm.retail.AEF.action.ACELogonActionImpl](#)
- ManagerOverrideAction=[com.ibm.retail.AEF.action.ACEManagerOverrideActionImpl](#)
- OperatorOverrideAction=[com.ibm.retail.AEF.action.ACEOperatorOverrideActionImpl](#)
- OverrideErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperOverride
- PrinterErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperPrinter
- PromptErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperPrompt
- RemoveReceiptErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperRemoveReceipt
- RetrieveTransaction=[com.ibm.retail.AEF.action.ACERetrieveTransactionActionImpl](#)
- RetrieveSuspendedTransactionList=[com.ibm.retail.AEF.action.ACERetrieveSuspendedTransactionListActionImpl](#)
- ACEErrorHelperB004=com.ibm.retail.AEF.util.ACEErrorHelperB004
- ACEErrorHelperB005=com.ibm.retail.AEF.util.ACEErrorHelperB005
- ACEErrorHelperB013=com.ibm.retail.AEF.util.ACEErrorHelperB013
- ACEErrorHelperB040=com.ibm.retail.AEF.util.ACEErrorHelperB040
- ACEErrorHelperB113=com.ibm.retail.AEF.util.ACEErrorHelperB113
- ACEErrorHelperB676=com.ibm.retail.AEF.util.ACEErrorHelperB676
- SignatureErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperSignature
- StartTransaction=[com.ibm.retail.AEF.action.ACEStartTransactionActionImpl](#)
- SuspendTransaction=[com.ibm.retail.AEF.action.ACESuspendTransactionActionImpl](#)
- TrainingModeOff=[com.ibm.retail.AEF.action.ACEForcedLogoffActionImpl](#)
- TrainingModeOn=[com.ibm.retail.AEF.action.ACESetTrainingModeOnActionImpl](#)
- VoidTransaction=[com.ibm.retail.AEF.action.ACEVoidTransactionActionImpl](#)
- VoucherNumberErrorHelper=com.ibm.retail.AEF.util.ACEErrorHelperVoucherNumber

Tender Actions

- AddAmericanExpressTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- AddAmexTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- AddCashTenderAction=com.ibm.retail.AEF.action.ACECashTenderActionImpl
- AddCheckTenderAction=com.ibm.retail.AEF.action.ACECheckTenderActionImpl
- AddCreditTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- AddCreditPlanATenderAction=
- AddCreditPlanBTenderAction=
- AddCreditPlanCTenderAction=
- AddCreditPlanDTenderAction=
- AddDiscoverTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- AddDueBillTenderAction=
- AddFoodStampsTenderAction=com.ibm.retail.AEF.action.ACEFoodStampsTenderActionImpl
- AddGiftCardTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- AddGiftCertificateTenderAction=com.ibm.retail.AEF.action.ACEGiftCertificateActionImpl
- AddMasterCardTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- AddMiscellaneousTenderAction=
- AddTravelersCheckTenderAction=
- AddType16TenderAction=
- AddType17TenderAction=
- AddType18TenderAction=
- AddType19TenderAction=
- AddType20TenderAction=
- AddType21TenderAction=
- AddType22TenderAction=
- AddType23TenderAction=
- AddType24TenderAction=
- AddType25TenderAction=
- AddType26TenderAction=
- AddType27TenderAction=
- AddType28TenderAction=
- AddType29TenderAction=
- AddType30TenderAction=
- AddType31TenderAction=
- AddType32TenderAction=
- AddType33TenderAction=
- AddType34TenderAction=
- AddType35TenderAction=
- AddType36TenderAction=
- AddStoreCouponTenderAction=
- AddVendorCouponTenderAction=
- AddVisaCreditTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidAmericanExpressTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidAmexTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidCashTenderAction=com.ibm.retail.AEF.action.ACECashTenderActionImpl
- VoidCheckTenderAction=com.ibm.retail.AEF.action.ACECheckTenderActionImpl
- VoidCreditTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidCreditPlanATenderAction=
- VoidCreditPlanBTenderAction=

- VoidCreditPlanCTenderAction=
- VoidCreditPlanDTenderAction=
- VoidDiscoverTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidDueBillTenderAction=
- VoidFoodStampsTenderAction=com.ibm.retail.AEF.action.ACEFoodStampsTenderActionImpl
- VoidGiftCardTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidGiftCertificateTenderAction=com.ibm.retail.AEF.action.ACEGiftCertificateActionImpl
- VoidMasterCardTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl
- VoidMiscellaneousTenderAction=
- VoidTravelersCheckTenderAction=
- VoidType16TenderAction=
- VoidType17TenderAction=
- VoidType18TenderAction=
- VoidType19TenderAction=
- VoidType20TenderAction=
- VoidType21TenderAction=
- VoidType22TenderAction=
- VoidType23TenderAction=
- VoidType24TenderAction=
- VoidType25TenderAction=
- VoidType26TenderAction=
- VoidType27TenderAction=
- VoidType28TenderAction=
- VoidType29TenderAction=
- VoidType30TenderAction=
- VoidType31TenderAction=
- VoidType32TenderAction=
- VoidType33TenderAction=
- VoidType34TenderAction=
- VoidType35TenderAction=
- VoidType36TenderAction=
- VoidStoreCouponTenderAction=
- VoidVendorCouponTenderAction=
- VoidVisaCreditTenderAction=com.ibm.retail.AEF.action.ACEEPCreditTenderActionImpl

Tender Line Item Implementation classes

- GiftCardTender=com.ibm.retail.AEF.automation.CreditTenderImpl

nextFile=userclass – This property is used to tell AEF the name of the next property file in the bundle chain.

Appclass.properties (GSA)

The appclass.properties file includes the following properties.

General Classes

- **AcceptOrVoidTender=com.ibm.retail.AEF.util.GSAErrorHelperAcceptOrVoidTender**
- **AccountNumberErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperAccountNumber**

- AEFErrorImpl=com.ibm.retail.AEF.util.GSAAEFErrorImpl
- AgeRestrictedHelper=com.ibm.retail.AEF.util.GSAAgeRestrictedHelper
- AppDefaultErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperDefault
- ApplInit=com.ibm.retail.AEF.util.GSAAppInit
- ApplyDelayedCouponsAction=com.ibm.retail.AEF.action.UnsupportedActionImpl
- BadConditionsImpl=[com.ibm.retail.AEF.util.GSABadConditionsImpl](#)
- BaseDescriptor449ErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperBaseDescriptor449
- CancelOverride=com.ibm.retail.AEF.action.UnsupportedActionImpl
- CancelPreviousEntryAction=[com.ibm.retail.AEF.action.GSACancelPreviousEntryActionImpl](#)
- ClearAction=[com.ibm.retail.AEF.action.GSAClearActionImpl](#)
- CustomerEntryAction=com.ibm.retail.AEF.action.UnsupportedActionImpl
- DescriptorsImpl=com.ibm.retail.AEF.util.GSADescriptorsImpl
- EnterAuthorizationErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperEnterAuthorization
- FatalErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperFatal
- ForcedLogoffAction=[com.ibm.retail.AEF.action.GSAForcedLogoffActionImpl](#)
- InformationalErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperInformational
- InsertDocumentHelper=com.ibm.retail.AEF.util.GSAErrorHelperInsertDocument
- ItemCodeParser=com.ibm.retail.AEF.automation.GSAItemCodeParserImpl
- ItemEntryAction=[com.ibm.retail.AEF.action.GSAItemEntryActionImpl](#)
- ItemReturnAction=[com.ibm.retail.AEF.action.GSAItemReturnActionImpl](#)
- ItemVoidAction=[com.ibm.retail.AEF.action.GSAItemVoidActionImpl](#)
- LogoffAction=[com.ibm.retail.AEF.action.GSALogoffActionImpl](#)
- LogonAction=[com.ibm.retail.AEF.action.GSALogonActionImpl](#)
- ManagerOverrideAction=[com.ibm.retail.AEF.action.GSAManagerOverrideActionImpl](#)
- OperatorOverrideAction=com.ibm.retail.AEF.action.UnsupportedActionImpl
- OriginalSalespersonRequiredErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperOriginalSalespersonRequired
- OverrideErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperOverride
- PriceRequiredErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperPriceRequired
- PrintBypassErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperPrintBypass
- PrinterErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperPrinter
- PrinterPaperOutErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperPrinterPaperOut
- QuantityErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperQuantityRequired
- QuantityOrPriceErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperQuantityOrPrice
- RemoveDocumentHelper=com.ibm.retail.AEF.util.GSAErrorHelperRemoveDocument
- RetrieveSuspendedTransactionList=[com.ibm.retail.AEF.action.GSARetrieveSuspendedTransactionListActionImpl](#)
- RetrieveTransaction=[com.ibm.retail.AEF.action.GSARetrieveTransactionActionImpl](#)
- StartTransaction=[com.ibm.retail.AEF.action.GSAStartTransactionActionImpl](#)
- SuspendTransaction=[com.ibm.retail.AEF.action.GSASuspendTransactionActionImpl](#)
- TerminalDisabledErrorHelper=com.ibm.retail.AEF.util.GSAErrorHelperTerminalDisabled
- TrainingModeOff=[com.ibm.retail.AEF.action.GSAForcedLogoffActionImpl](#)
- TrainingModeOn=[com.ibm.retail.AEF.action.GSASetTrainingModeOnActionImpl](#)
- VoidTransaction=[com.ibm.retail.AEF.action.GSAVoidTransactionActionImpl](#)

Tender Actions

- AddAmericanExpressTenderAction=[com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl](#)
- AddAmexTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- AddCashTenderAction=
- AddCheckTenderAction=
- AddCreditTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl

- AddCreditPlanATenderAction=
- AddCreditPlanBTenderAction=
- AddCreditPlanCTenderAction=
- AddCreditPlanDTenderAction=
- AddDinersClubTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- AddDueBillTenderAction=
- AddFoodStampsTenderAction=
- AddGiftCertificateTenderAction=
- AddMasterCardTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- AddMiscellaneousTenderAction=
- AddTravelersCheckTenderAction=
- AddType16TenderAction=
- AddType17TenderAction=
- AddType18TenderAction=
- AddType19TenderAction=
- AddType20TenderAction=
- AddType21TenderAction=
- AddType22TenderAction=
- AddType23TenderAction=
- AddType24TenderAction=
- AddType25TenderAction=
- AddType26TenderAction=
- AddType27TenderAction=
- AddType28TenderAction=
- AddType29TenderAction=
- AddType30TenderAction=
- AddType31TenderAction=
- AddType32TenderAction=
- AddType33TenderAction=
- AddType34TenderAction=
- AddType35TenderAction=
- AddType36TenderAction=
- AddStoreCouponTenderAction=
- AddVendorCouponTenderAction=
- AddVisaCreditTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- VoidAmericanExpressTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- VoidAmexTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- VoidCashTenderAction=
- VoidCheckTenderAction=
- VoidCreditTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- VoidCreditPlanATenderAction=
- VoidCreditPlanBTenderAction=
- VoidCreditPlanCTenderAction=
- VoidCreditPlanDTenderAction=
- VoidDinersClubTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- VoidDueBillTenderAction=
- VoidFoodStampsTenderAction=
- VoidGiftCertificateTenderAction=
- VoidMasterCardTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl
- VoidMiscellaneousTenderAction=

- VoidTravelersCheckTenderAction=
- VoidType16TenderAction=
- VoidType17TenderAction=
- VoidType18TenderAction=
- VoidType19TenderAction=
- VoidType20TenderAction=
- VoidType21TenderAction=
- VoidType22TenderAction=
- VoidType23TenderAction=
- VoidType24TenderAction=
- VoidType25TenderAction=
- VoidType26TenderAction=
- VoidType27TenderAction=
- VoidType28TenderAction=
- VoidType29TenderAction=
- VoidType30TenderAction=
- VoidType31TenderAction=
- VoidType32TenderAction=
- VoidType33TenderAction=
- VoidType34TenderAction=
- VoidType35TenderAction=
- VoidType36TenderAction=
- VoidStoreCouponTenderAction=
- VoidVendorCouponTenderAction=
- VoidVisaCreditTenderAction=com.ibm.retail.AEF.action.GSAEFTCreditTenderActionImpl

nextFile=userclass – This property is used to tell AEF the name of the next property file in the bundle chain.

Appclass.properties (SA)

The appclass.properties file includes the following properties.

General Classes

- AccountNumberErrorHandler=com.ibm.retail.AEF.util.SAErrorHandlerAccountNumber
- AEFErrorImpl=com.ibm.retail.AEF.util.SAAEFErrorImpl
- AgeRestrictedHelper=com.ibm.retail.AEF.util.SAAgeRestrictedHelper
- AppDefaultErrorHandler=com.ibm.retail.AEF.util.SAErrorHandlerDefault
- ApplInit=com.ibm.retail.AEF.util.SAApplInit
- ApplyDelayedCouponsAction=[com.ibm.retail.AEF.action.SAApplDelayedCouponsActionImpl](#)
- BadConditionsImpl=[com.ibm.retail.AEF.util.SABadConditionsImpl](#)
- CancelOverride=[com.ibm.retail.AEF.action.SACancelOverrideActionImpl](#)
- CancelPreviousEntryAction=[com.ibm.retail.AEF.action.SACancelPreviousEntryActionImpl](#)
- ClearAction=[com.ibm.retail.AEF.action.SAClearActionImpl](#)
- CustomerEntryAction=[com.ibm.retail.AEF.action.SACustomerEntryActionImpl](#)
- DescriptorsImpl=com.ibm.retail.AEF.util.SADescriptorsImpl
- DuplicateReceiptHelper=com.ibm.retail.AEF.util.SAErrorHandlerDuplicateReceipt
- EnterStandAlone=[com.ibm.retail.AEF.action.SAEnterStandAloneActionImpl](#)
- FatalClearErrorHandler=com.ibm.retail.AEF.util.SAErrorHandlerFatalClear
- FatalErrorHandler=com.ibm.retail.AEF.util.SAErrorHandlerFatal
- ForcedLogoffAction=[com.ibm.retail.AEF.action.SAForcedLogoffActionImpl](#)
- IgnoreErrorHandler=com.ibm.retail.AEF.util.SAErrorHandlerIgnore

- InformationalErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperInformational
- ItemEntryAction=[com.ibm.retail.AEF.action.SAItemEntryActionImpl](#)
- ItemPromptErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperItemPrompt
- ItemReturnAction=[com.ibm.retail.AEF.action.SAItemReturnActionImpl](#)
- ItemVoidAction=[com.ibm.retail.AEF.action.SAItemVoidActionImpl](#)
- LevelOverrideErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperLevelOverride
- LogoffAction=[com.ibm.retail.AEF.action.SALogoffActionImpl](#)
- LogonAction=[com.ibm.retail.AEF.action.SALogonActionImpl](#)
- ManagerOverrideAction=[com.ibm.retail.AEF.action.SAManagerOverrideActionImpl](#)
- OperatorOverrideAction=[com.ibm.retail.AEF.action.SAOperatorOverrideActionImpl](#)
- OverrideErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperOverride
- PrinterErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperPrinter
- PrinterPaperOutErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperPrinterPaperOut
- PromptErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperPrompt
- RemoveReceiptErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperRemoveReceipt
- RetrieveTransaction=[com.ibm.retail.AEF.action.SARetrieveTransactionActionImpl](#)
- RetrieveSuspendedTransactionList=[com.ibm.retail.AEF.action.SARetrieveSuspendedTransactionListActionImpl](#)
- SAErrorHelperB004=com.ibm.retail.AEF.util.SAErrorHelperB004
- SAErrorHelperB005=com.ibm.retail.AEF.util.SAErrorHelperB005
- SAErrorHelperB013=com.ibm.retail.AEF.util.SAErrorHelperB013
- SAErrorHelperB040=com.ibm.retail.AEF.util.SAErrorHelperB040
- SAErrorHelperB113=com.ibm.retail.AEF.util.SAErrorHelperB113
- SAErrorHelperB537=com.ibm.retail.AEF.util.SAErrorHelperB537
- SAErrorHelperB608=com.ibm.retail.AEF.util.SAErrorHelperB608
- SAErrorHelperB676=com.ibm.retail.AEF.util.SAErrorHelperB676
- SignatureErrorHelper=com.ibm.retail.AEF.util.SAErrorHelperSignature
- StartTransaction=[com.ibm.retail.AEF.action.SAStartTransactionActionImpl](#)
- SuspendTransaction=[com.ibm.retail.AEF.action.SASuspendTransactionActionImpl](#)
- TrainingModeOff=[com.ibm.retail.AEF.action.SAForcedLogoffActionImpl](#)
- TrainingModeOn=[com.ibm.retail.AEF.action.SASetTrainingModeOnActionImpl](#)
- VoidTransaction=[com.ibm.retail.AEF.action.SAVoidTransactionActionImpl](#)

Tender Actions

- AddAmericanExpressTenderAction=[com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl](#)
- AddAmexTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- AddCashTenderAction=com.ibm.retail.AEF.action.SACashTenderActionImpl
- AddCheckTenderAction=com.ibm.retail.AEF.action.SACheckTenderActionImpl
- AddCreditTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- AddCreditCardTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- AddDinersClubTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- AddFoodStampsTenderAction=com.ibm.retail.AEF.action.SAFoodStampsTenderActionImpl
- AddGiftCertificateTenderAction=com.ibm.retail.AEF.action.SAGiftCertificateActionImpl
- AddMasterCardTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- AddMiscellaneousTenderAction=
- AddTravelersCheckTenderAction=
- AddStoreCouponTenderAction=
- AddVendorCouponTenderAction=

- AddVisaCreditTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidAmericanExpressTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidAmexTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidCashTenderAction=com.ibm.retail.AEF.action.SACashTenderActionImpl
- VoidCheckTenderAction=com.ibm.retail.AEF.action.SACheckTenderActionImpl
- VoidCreditTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidCreditCardTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidDinersClubTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidFoodStampsTenderAction=com.ibm.retail.AEF.action.SAFoodStampsTenderActionImpl
- VoidGiftCertificateTenderAction=com.ibm.retail.AEF.action.SAGiftCertificateActionImpl
- VoidMasterCardTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl
- VoidMiscellaneousTenderAction=
- VoidTravelersCheckTenderAction=
- VoidStoreCouponTenderAction=
- VoidVendorCouponTenderAction=
- VoidVisaCreditTenderAction=com.ibm.retail.AEF.action.SAEFTCreditTenderActionImpl

nextFile=userclass – This property is used to tell AEF the name of the next property file in the bundle chain.

Configuration Bundle

This bundle contains configuration properties for the AEF. These properties include settings for AEFBASE and the SessionServer objects. The files that make up the Configuration bundle are config.properties, appconfig.properties, and userconfig.properties.

For each property, a key is provided followed by an equal sign and a value. If the value is %nodename%, then the value will be evaluated to the controller ID or terminal number where the code is running. If the value is %platformSpecific%, then AEF will process the value based on what platform the code is running on.

Config.properties

The config.properties file includes the following properties.

- **mgmt.enable=%platformSpecific%** - If mgmt.enable is true, the JMX GeneralAgent object is created as well as various AEF MBeans
- **4690terminal.mgmt.enable=true**
- **4690css.mgmt.enable=true**
- **general.mgmt.enable=false**
- **create.server=%platformSpecific%** - If create.server is true, the SessionServer object is created.
- **4690terminal.create.server=true**
- **4690css.create.server=true**
- **general.create.server=false**
- **real.session.wait=60000**

- **server.id=AEF_SESSION_SERVER%nodename%** - RMI registry ID for the SessionServer object.
- **server.remote.access=%platformSpecific%** - If server.remote.access is true, the SessionServer object is registered in the RMI registry with the rmi.port and server.id values as follows: rmi://:1099/AEF_SESSION_SERVER. If server.remote.access is set to "%platformSpecific%", then the server will be registered with the RMI registry depending on the platform specific property. For example, if the session server is running in the 4690 css jvm, then the property "4690css.server.remote.access" property controls whether the session server is registered with RMI for remote access.
- **4690terminal.server.remote.access=false**
- **4690css.server.remote.access=true**
- **general.server.remote.access=true**
- **create.factory=true** - Create the factory. Factories may only be created on 4690 terminals or controllers, so this property is ignored when running on non-4690 platforms.
- **factory.remote.access=%platformSpecific%** - Allow remote access to the factory. Ignored if create.factory is false. Values for factory.remote.access may be "true", "false", or "%platformSpecific%". If set to "%platformSpecific%", then the appropriate property will be used depending on whether the factory is for a terminal, or for CSS. Note that if there is no need to gain remote access to real terminal sessions, some performance savings could be realized by setting 4690terminal.factory.remote.access to false.
- **4690terminal.factory.remote.access=true**
- **4690css.factory.remote.access=true**
- **factory.beacon.interval=%platformSpecific%** - Determines how often (in milliseconds) a factory with remote access should beacon its presence. SessionServers listen for this beacon to establish contact with remote AEFSessionFactories. Value is ignored if create.factory or factory.remote.access is false.
- **4690terminal.factory.beacon.interval=60000**
- **4690css.factory.beacon.interval=15000**
- **multicast.address=229.107.3.145** - Specifies the IP address for the multicast socket used for the factory beacon. This should only ever need to be changed if the address was already in use by another application. Note that this address does not correspond to a physical IP address of any machine in the network. It must be in the multicast range of 224.0.0.1 to 239.255.255.255 inclusive.
- **multicast.port=5445** - Specifies the IP port for the multicast socket used for the factory beacon. This should only ever need to be changed if the port was already in use by another application.
- **create.rmi.registry=%platformSpecific%** - Create the RMI registry on the rmi.port port. If rmi.port is set to "%platformSpecific%", then either the value for 4690.controller.rmi.port or general.rmi.port will be used depending on whether the AEF is being run on a 4690 controller. On a 4690 controller, the value from 4690.controller.rmi.port will be used. On any other platform (including the 4690 terminal) the value from general.rmi.port will be used. Note: If there is no need for remote access to terminal factories or SessionServers, then some performance savings can be realized by setting 4690terminal.create.rmi.registry to false.
- **4690terminal.create.rmi.registry=true**
- **4690css.create.rmi.registry=true**
- **general.create.rmi.registry=false**
- **rmi.port=%platformSpecific%**
- **4690.controller.rmi.port=12099**
- **general.rmi.port=11099**
- **rmi.timeout=90000**

- **rmi.leaseValue=60000**
- **rmi.checkInterval=30000**
- **health.port=%platformSpecific%** - Specifies the IP port for the server socket used for the AEF Health service. This should only ever need to be changed if the port was already in use by another application.
- **4690.controller.health.port=12199**
- **general.health.port=11199**
- **health.server.timeout=60000** - this is the socket timeout value for listening to client pings.
- **health.server.retry.limit=5** - This is the number of consecutive timeouts allowed before stopping the client listener thread. The client is considered to be unreachable.
- **health.monitor.interval=3000** - This is the interval or wait time between monitor pings to the server
- **health.monitor.timeout=2000** - This is the amount of time the monitor will wait for a response from the health server before firing a healthCheckFailed event to listeners.
- **default.read.timeout=160000** - RMI timeout values for Remote objects (milliseconds, 0=no timeout). This is the default read timeout for all remote objects.
- **default.connect.timeout=160000** - RMI timeout values for Remote objects (milliseconds, 0=no timeout). This is the default connect timeout for all remote objects.
- **com.ibm.retail.AEF.client.AEFEventListenerProxy.read.timeout=5000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is value used for SO_TIMEOUT
- **com.ibm.retail.AEF.client.AEFEventListenerProxy.connect.timeout=5000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is the value used for connect() timeout.
- **com.ibm.retail.AEF.client.KeyConsumerProxy.read.timeout=5000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is value used for SO_TIMEOUT
- **com.ibm.retail.AEF.client.KeyConsumerProxy.connect.timeout=5000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is the value used for connect() timeout.
- **com.ibm.retail.AEF.workstation.WorkstationImpl.read.timeout=5000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is value used for SO_TIMEOUT
- **com.ibm.retail.AEF.workstation.WorkstationImpl.connect.timeout=5000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is the value used for connect() timeout.
- **com.ibm.retail.AEF.automation.POSAutomationProviderImpl.read.timeout=160000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is value used for SO_TIMEOUT
- **com.ibm.retail.AEF.automation.POSAutomationProviderImpl.connect.timeout=160000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is the value used for connect() timeout
- **com.ibm.retail.AEF.automation.TransactionImpl.read.timeout=160000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is value used for SO_TIMEOUT
- **com.ibm.retail.AEF.automation.TransactionImpl.connect.timeout=160000** - RMI timeout values for specific classes (milliseconds, 0=no timeout). This is the value used for connect() timeout.
- **factory.url.0=local** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.1=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)

- **factory.url.2=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.3=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.4=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.5=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.6=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.7=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.8=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.url.9=** - AEFSessionFactory URLs (Use "local" for single JVM config). Supports 10 factories (0-9)
- **factory.id=AEF_SESSION_FACTORY%nodename%** - Configure AEF settings for local AEFSessionFactory.
- **ready.wait.timeout=120000** - Configure AEF settings for local AEFSessionFactory.
- **event.queue.timeout=3000** - Configure AEF settings for local AEFSessionFactory.
- **server.poll.interval=10000** - Configure AEF settings for local AEFSessionFactory.
- **logger.remote.access=%platformSpecific%** - LoggerControl properties.
- **4690terminal.logger.remote.access=false** - LoggerControl properties.
- **4690css.logger.remote.access=true** - LoggerControl properties.
- **general.logger.remote.access=false** - LoggerControl properties.
- **default.encoding=UTF-8** - LoggerControl properties.
- **create.device.server=false** - DeviceServer properties. The device server is needed only for sessions with remotely attached devices.
- **device.server.port=-1** - DeviceServer properties. The device server is needed only for sessions with remotely attached devices.
- **base.extensions=** - Configure AEF settings for base level (per JVM). The AEF will instantiate an instance of each class listed (comma separated) Example:
base.extensions=com.xyz.foo.BaseFoo1,com.xyz.foo.BaseFoo2. Instances of BaseFoo1 and BaseFoo2 will be instantiated per AEFSessionFactory (1 per JVM).
NOTES: Extension class must use a default constructor for instantiation. 2. Base extension objects are created during AEFSessionFactory initialization
- **debug.css.pos.appl=false** - If you need to run the 4680-4690 debugger on an application such as IBM 4680-4690 Supermarket Application or IBM 4680-4690 General Sales Application which is running under CSS on a pure controller, set this parameter to true.
- **ItemSalesListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in classes.properties.
- **CashReceiptListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in classes.properties.
- **TenderListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will

create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.

- **CouponListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **CustomerListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **OperatorListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **DiscountListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **POSAppEventListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **PointsListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **OptionsListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **StateChangeListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.
- **WorkstationStatusListeners=** - POSDataProvider property change listeners. Each listener type is followed by a list of classname keys separated by commas if there is more than one listener to be created for the type. Upon initialization, the POSDataProvider will create an instance of each class listed, and add it as a listener. Note that the class name keys are used to look up the fully qualified classname in `classes.properties`.

- **TransactionTotalsListeners=TransactionTotalsSetter** - TransactionTotalsSetter receives each of the TransactionTotalsEvents and populates the SalesTransaction's TransactionTotals object.
- **TransactionStatusListeners=**
- **KEY_SEQUENCE_TIMEOUT=60000** - This is the number of milliseconds to wait for some condition to happen after a keysequence is sent before timing out.
- **PAYMENT_HOST_SEQUENCE_TIMEOUT=180000** - This is the number of milliseconds to wait for some condition to happen after a keysequence is sent before timing out. This timeout is used in sequences that will generate a round trip to a payment host. The wait time is typically longer than for normal key sequences.
- **ERROR_HANDLER_SLEEP_INTERVAL_1=-1** - This is the number of milliseconds to wait before continuing with the next operation while handling errors. During testing, we found that differences in hardware, and software configuration could lead to timing issues. In many cases, invalid key sequence errors were erroneously generated. In order to compensate for this, JAVA sleep calls were added to the ErrorHandler classes.
- **ERROR_HANDLER_SLEEP_INTERVAL_2=-1** - This is the number of milliseconds to wait before continuing with the next operation while handling errors. During testing, we found that differences in hardware, and software configuration could lead to timing issues. In many cases, invalid key sequence errors were erroneously generated. In order to compensate for this, JAVA sleep calls were added to the AEFErrorHandler classes.
- **ACTION_PROCESSOR_THREADS=30** - Determines the number of threads in the action processor thread pool. This is the number of concurrent actions which may be performed by the automation provider. In general, each function in the automation provider will require one action to perform the method. The default value is 5, which means that there may be 5 concurrent requests for methods in the automation provider. If all 5 thread pool threads are in use, any additional requests would have to wait for one of the 5 threads to free up.
- **LOCALE_LANGUAGE=en** - The language argument is a valid ISO Language Code. These codes are the lower-case, two-letter codes as defined by ISO-639.
- **LOCALE_COUNTRY=US** - The country argument is a valid ISO Country Code. These codes are the upper-case, two-letter codes as defined by ISO-3166.
- **create.memory.debug=true** - Create in memory debug.
- **max.retries.for.locked.input.queue=120** - Determines the maximum number of retries the AEF will do while attempting to perform a key sequence while the input queue is locked. If the maximum number of retries is reached, and the input queue is still locked, the POSAutomationProvider command will throw an exception with an error code of AEFCnst.SYSTEM_BUSY.
- **locked.input.queue.delay=250** - Determines the amount of time to wait (in milliseconds) in between each attempt to perform a key sequence while the input queue is locked.

nextFile=appconfig - This property is used to tell AEF the name of the next property file in the bundle chain.

Appconfig.properties (ACE)

The appconfig.properties file includes the following properties.

- **APPLICATION_DATE_PATTERN=yyMMdd** - Determine how the date sent from the POS application as part of the TransactionStatusEvent are parsed. See the Java JDK documentation for the SimpleDateFormat class for details on the valid pattern values.

- **APPLICATION_TIME_PATTERN=kk:mm** - Determine how the time sent from the POS application as part of the TransactionStatusEvent are parsed. See the Java JDK documentation for the SimpleDateFormat class for details on the valid pattern values.
- **CREDIT_APPROVED_MESSAGE=Approved** - Specifies the message which is displayed on the 2x20 for an EFT credit approval.
- **CREDIT_LOCAL_APPROVED_MESSAGE=B309** - Specifies the message which is displayed on the 2x20 for an EFT credit approval.

nextFile=userconfig - This property is used to tell AEF what the next file in the bundle is.

Appconfig.properties (GSA)

The appconfig.properties file includes the following properties.

- **APPLICATION_DATE_PATTERN=yyMMdd** - Determine how the date sent from the POS application as part of the TransactionStatusEvent are parsed. See the Java JDK documentation for the SimpleDateFormat class for details on the valid pattern values.
- **APPLICATION_TIME_PATTERN=kk:mm** - Determine how the time sent from the POS application as part of the TransactionStatusEvent are parsed. See the Java JDK documentation for the SimpleDateFormat class for details on the valid pattern values.

nextFile=userconfig - This property is used to tell AEF the name of the next property file in the bundle chain.

Appconfig.properties (SA)

The appconfig.properties file includes the following properties.

- **APPLICATION_DATE_PATTERN=yyMMdd** - Determine how the date sent from the POS application as part of the TransactionStatusEvent are parsed. See the Java JDK documentation for the SimpleDateFormat class for details on the valid pattern values.
- **APPLICATION_TIME_PATTERN=kk:mm** - Determine how the time sent from the POS application as part of the TransactionStatusEvent are parsed. See the Java JDK documentation for the SimpleDateFormat class for details on the valid pattern values.
- **CREDIT_APPROVED_MESSAGE=B531** - Specifies the message which is displayed on the 2x20 for an EFT credit approval.

○ **nextFile=userconfig** - This property is used to tell AEF the name of the next property file in the bundle chain.

Error Bundle

The Error bundle is used to specify what error helpers should be used to handle specific errors. The files that makeup the Error bundle are error.properties, apperror.properties and usererror.properties.

Each error helper has four keys associated with each entry. These keys are ADDITIONAL_TEXT, HANDLING_CLASS_KEY, ERROR_CODE, and EXTENDED_ERROR_CODE. The ADDITIONAL_TEXT is used to provide the operator

with additional information on the error. This can include information on how to clear the error or what information to collect to help in problem determination. The HANDLING_CLASS_KEY is used to create the error helper. The value of this key must be a key in the Class bundle. The ERROR_CODE, and EXTENDED_ERROR_CODE values are used as return values to the calling application to give an indication as to what errors were encountered. The value of these keys must be constants in AEFCnst.java.

Error.properties

The error.properties file includes the following properties.

- Default error helper used when the key is not found in the error bundle or no helper class is specified.
 - DEFAULT_ADDITIONAL_TEXT=The Default error helper was used because this error or its helper class is not defined in the error property files.
 - DEFAULT_HANDLING_CLASS_KEY=DefaultErrorHandler
 - DEFAULT_ERROR_CODE=ERROR_HANDLER_OTHER_ERROR
 - DEFAULT_EXTENDED_ERROR_CODE=NONE

nextFile=apperror – This property is used to tell AEF the name of the next property file in the bundle chain.

Apperror.properties (ACE)

The apperror.properties file includes the following properties.

- Default error helper used when the key is not found in the error bundle or no helper class is specified.
 - APP_DEFAULT_ADDITIONAL_TEXT=The Default error helper was used because this error or its helper class is not defined in the error property files.
 - APP_DEFAULT_HANDLING_CLASS_KEY=AppDefaultErrorHandler
 - APP_DEFAULT_ERROR_CODE=ERROR_HANDLER_OTHER_ERROR
 - APP_DEFAULT_EXTENDED_ERROR_CODE=NONE
- B000 ABNORMAL APPLICATION END
 - B000_ADDITIONAL_TEXT=An unrecoverable terminal error has been detected.
 - B000_HANDLING_CLASS_KEY=FatalErrorHandler
 - B000_ERROR_CODE=POS_APP_FAILURE
 - B000_EXTENDED_ERROR_CODE=NONE
- B001 NO COUPON ON
 - B001_ADDITIONAL_TEXT=You pressed the MFR COUPON or STORE COUPON key for an item code that cannot be used as a coupon, or for a department key that cannot be used with coupon keys.
 - B001_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B001_ERROR_CODE=INVALID_ARGUMENT
 - B001_EXTENDED_ERROR_CODE=INVALID_ITEM_CODE
- B002 SYSTEM IS BUSY PLEASE WAIT
 - B002_ADDITIONAL_TEXT=The system has not finished processing previous entries. The entry was not accepted by the system.
 - B002_HANDLING_CLASS_KEY=FatalErrorHandler
 - B002_ERROR_CODE=SYSTEM_BUSY
 - B002_EXTENDED_ERROR_CODE=NONE
- B003 CHECK KEY SEQUENCE

- B003_ADDITIONAL_TEXT=You used a key sequence that is invalid or that cannot be used at this time. For example, you will get this error if you attempt to void a tender without having entered it.
 - B003_HANDLING_CLASS_KEY=FatalErrorHandler
 - B003_ERROR_CODE=CONFIG_ERROR
 - B003_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B004 MANAGER'S KEY IS REQUIRED
 - B004_ADDITIONAL_TEXT=A manager's key is required to perform a manager override.
 - B004_HANDLING_CLASS_KEY=ACEErrorHandlerB004
 - B004_ERROR_CODE=KEYLOCK_ERROR
 - B004_EXTENDED_ERROR_CODE=MANAGER_KEY_REQUIRED
- B005 REMOVE MANAGER'S KEY
 - B005_ADDITIONAL_TEXT=The manager's key must be removed after being used with a manager override sequence.
 - B005_HANDLING_CLASS_KEY=ACEErrorHandlerB005
 - B005_ERROR_CODE=KEYLOCK_ERROR
 - B005_EXTENDED_ERROR_CODE=MANAGER_KEY_NOT_REMOVED
- B006 KEYED DATA OUT OF RANGE
 - B006_ADDITIONAL_TEXT=A numeric entry in the key sequence is too large or too small.
 - B006_HANDLING_CLASS_KEY=FatalErrorHandler
 - B006_ERROR_CODE=INVALID_ARGUMENT
 - B006_EXTENDED_ERROR_CODE=DATA_OUT_OF_RANGE
- B007 CHECK THE PRINTER
 - B007_ADDITIONAL_TEXT=An error occurred while trying to print a line or a terminal without an attached printer was powered on.
 - B007_HANDLING_CLASS_KEY=PrinterErrorHandler
 - B007_ERROR_CODE=PRINTER_ERROR
 - B007_EXTENDED_ERROR_CODE=NONE
- B008 %1 TRANSACTION LIMIT CHECK
 - B008_ADDITIONAL_TEXT=A transaction limit has been exceeded:
 - B008_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B008_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B008_EXTENDED_ERROR_CODE=TRANSACTION_LIMIT
- B009 %1 ITEM LIMIT CHECK
 - B009_ADDITIONAL_TEXT=An item limit has been exceeded:
 - B009_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B009_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B009_EXTENDED_ERROR_CODE=ITEM_LIMIT
- B010 USE LOOKUP KEYS
 - B010_ADDITIONAL_TEXT=The Item Record file cannot be successfully accessed.
 - B010_HANDLING_CLASS_KEY=FatalErrorHandler
 - B010_ERROR_CODE=POS_APP_FAILURE
 - B010_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B011 CHECK COIN DISPENSER
 - B011_ADDITIONAL_TEXT=
 - B011_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B011_ERROR_CODE=COIN_DISPENSER_ERROR
 - B011_EXTENDED_ERROR_CODE=NONE
- B012 CHECK JOURNAL PAPER
 - B012_ADDITIONAL_TEXT=A paper-low condition has been detected on the Transaction Summary Journal.
 - B012_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B012_ERROR_CODE=PRINTER_ERROR

- B012_EXTENDED_ERROR_CODE=PAPER_LOW
- B013 CLOSE THE CASH DRAWER
 - B013_ADDITIONAL_TEXT=The cash drawer must be closed at the start of a transaction and within a user-specified time after the end of a transaction.
 - B013_HANDLING_CLASS_KEY=ACEErrorHelperB013
 - B013_ERROR_CODE=CASH_DRAWER_ERROR
 - B013_EXTENDED_ERROR_CODE=CLOSE_DRAWER
- B014 NO TRANSACTION RECOVERY
 - B014_ADDITIONAL_TEXT=A transaction in process has not been recovered following a terminal restart or terminal transfer.
 - B014_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B014_ERROR_CODE=NONE
 - B014_EXTENDED_ERROR_CODE=NONE
- B015 CHECK THE KEYBOARD
 - B015_ADDITIONAL_TEXT=The application program failed in an attempt to reach the beeper in the keyboard.
 - B015_HANDLING_CLASS_KEY=FatalErrorHelper
 - B015_ERROR_CODE=KEYBOARD_ERROR
 - B015_EXTENDED_ERROR_CODE=NONE
- B016 WRITE ERROR TRANSACTION LOST
 - B016_ADDITIONAL_TEXT=A hardware error caused a write failure in the Transaction Summary Log and the completed transaction is lost.
 - B016_HANDLING_CLASS_KEY=FatalErrorHelper
 - B016_ERROR_CODE=POS_APP_FAILURE
 - B016_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B017 EXTENDED PRICE LIMIT CHECK
 - B017_ADDITIONAL_TEXT=The product of quantity and price cannot exceed eight digits for any item entry.
 - B017_HANDLING_CLASS_KEY=FatalErrorHelper
 - B017_ERROR_CODE=INVALID_ARGUMENT
 - B017_EXTENDED_ERROR_CODE=EXTENDED_PRICE_TOO_LARGE
- B018 NO WEIGHT ON
 - B018_ADDITIONAL_TEXT=The WEIGHT or TARE key was pressed for an item that is not sold by weight.
 - B018_HANDLING_CLASS_KEY=FatalErrorHelper
 - B018_ERROR_CODE=INVALID_ARGUMENT
 - B018_EXTENDED_ERROR_CODE=ITEM_WEIGHT_PROHIBITED
- B019 WEIGHT NEEDED
 - B019_ADDITIONAL_TEXT=A weight or price must be entered for an item that is sold by weight.
 - B019_HANDLING_CLASS_KEY=ItemPromptErrorHelper
 - B019_ERROR_CODE=INVALID_ARGUMENT
 - B019_EXTENDED_ERROR_CODE=ITEM_WEIGHT_REQUIRED
- B020 TAKE TOTAL
 - B020_ADDITIONAL_TEXT=A transaction total is required: Between an item sale and a tender; Between an item sale and a scanned UPC category coupon; Between a scanned UPC category 5 coupon and an item sale; Between a food stamp balance and an item sale; Between a food stamp tender and an item sale; Before discount or discount cancellation; Before a food stamp balance
 - B020_HANDLING_CLASS_KEY=FatalErrorHelper
 - B020_ERROR_CODE=CONFIG_ERROR
 - B020_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B021 CHECK TARE CODE
 - B021_ADDITIONAL_TEXT=The tare code entered with the TARE key is not defined.

- B021_HANDLING_CLASS_KEY=FatalErrorHandler
 - B021_ERROR_CODE=INVALID_ARGUMENT
 - B021_EXTENDED_ERROR_CODE=INVALID_TARE
- B022 NO DEPOSIT ON
 - B022_ADDITIONAL_TEXT=The DEPOSIT key has been pressed for an item code that cannot be used as a deposit.
 - B022_HANDLING_CLASS_KEY=FatalErrorHandler
 - B022_ERROR_CODE=INVALID_ARGUMENT
 - B022_EXTENDED_ERROR_CODE=INVALID_ITEM_CODE
- B023 NO QUANTITY ON
 - B023_ADDITIONAL_TEXT=The QTY key was pressed for an item which is sold by weight or does not allow for quantity entry.
 - B023_HANDLING_CLASS_KEY=FatalErrorHandler
 - B023_ERROR_CODE=INVALID_ARGUMENT
 - B023_EXTENDED_ERROR_CODE=ITEM_QUANTITY_PROHIBITED
- B024 PRICE NEEDED
 - B024_ADDITIONAL_TEXT=A price must be entered for this item.
 - B024_HANDLING_CLASS_KEY=ItemPromptErrorHandler
 - B024_ERROR_CODE=INVALID_ARGUMENT
 - B024_EXTENDED_ERROR_CODE=ITEM_PRICE_REQUIRED
- B025 SELL ITEM WITHOUT REPEAT KEY
 - B025_ADDITIONAL_TEXT=The limit for consecutive repeat entries using a repeat key has been exceeded.
 - B025_HANDLING_CLASS_KEY=FatalErrorHandler
 - B025_ERROR_CODE=CONFIG_ERROR
 - B025_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B026 ITEM NOT FOUND
 - B026_ADDITIONAL_TEXT=The displayed item code is not on the Item Record file. This could be the item code being sold or a linked item.
 - B026_HANDLING_CLASS_KEY=FatalErrorHandler
 - B026_ERROR_CODE=ITEM_NOT_FOUND
 - B026_EXTENDED_ERROR_CODE=NONE
- B027 NOT FOR SALE
 - B027_ADDITIONAL_TEXT=The item is not for sale or is not allowed to be sold during selected time periods.
 - B027_HANDLING_CLASS_KEY=FatalErrorHandler
 - B027_ERROR_CODE=ITEM_NOT_FOR_SALE
 - B027_EXTENDED_ERROR_CODE=NONE
- B028 TAKE FOODSTAMP TOTAL
 - B028_ADDITIONAL_TEXT=A food stamp total is required before a food stamp tender.
 - B028_HANDLING_CLASS_KEY=FatalErrorHandler
 - B028_ERROR_CODE=CONFIG_ERROR
 - B028_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B029 QUANTITY NEEDED
 - B029_ADDITIONAL_TEXT=A quantity must be entered for this item.
 - B029_HANDLING_CLASS_KEY=ItemPromptErrorHandler
 - B029_ERROR_CODE=INVALID_ARGUMENT
 - B029_EXTENDED_ERROR_CODE=ITEM_QUANTITY_REQUIRED
- B030 TRANSACTION TOTAL IS TOO LARGE
 - B030_ADDITIONAL_TEXT=The balance due has exceeded the 9-digit upper limit of the system.
 - B030_HANDLING_CLASS_KEY=FatalErrorHandler
 - B030_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B030_EXTENDED_ERROR_CODE=TRANSACTION_TOTAL_TOO_LARGE

- B031 NO F.S. KEY ON
 - B031_ADDITIONAL_TEXT=The FS/NO FS or FOOD STAMP keys are only allowed with selected department keys.
 - B031_HANDLING_CLASS_KEY=FatalErrorHandler
 - B031_ERROR_CODE=CONFIG_ERROR
 - B031_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B032 NO TAX KEY ON
 - B032_ADDITIONAL_TEXT=The TAX/NO TAX key is allowed only with selected department keys.
 - B032_HANDLING_CLASS_KEY=FatalErrorHandler
 - B032_ERROR_CODE=CONFIG_ERROR
 - B032_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B033 SCALE ERROR CLEAR/RETRY
 - B033_ADDITIONAL_TEXT=The scale driver detected bad data from the interface, or a read of the interface occurred.
 - B033_HANDLING_CLASS_KEY=FatalErrorHandler
 - B033_ERROR_CODE=SCALE_ERROR
 - B033_EXTENDED_ERROR_CODE=NONE
- B034 CHECK TENDER VARIETY
 - B034_ADDITIONAL_TEXT=One of the following conditions exists: 1) The tender variety entered with the asterisk (*) key is not defined. 2) Tender entered for a negative transaction is defined as not refundable.
 - B034_HANDLING_CLASS_KEY=FatalErrorHandler
 - B034_ERROR_CODE=CONFIG_ERROR
 - B034_EXTENDED_ERROR_CODE=UNDEFINED_TENDER_VARIETY
- B035 CHECK TENDER DENOMINATION
 - B035_ADDITIONAL_TEXT=For the tendered tender ID, the Denomination Test option in Options->Tender->Tenders personalization has been selected, which specifies that only dollar amounts can be tendered .
 - B035_HANDLING_CLASS_KEY=FatalErrorHandler
 - B035_ERROR_CODE=INVALID_ARGUMENT
 - B035_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- B036 CHECK QUANTITY
 - B036_ADDITIONAL_TEXT=The displayed quantity may be incorrect.
 - B036_HANDLING_CLASS_KEY=FatalErrorHandler
 - B036_ERROR_CODE=INVALID_ARGUMENT
 - B036_EXTENDED_ERROR_CODE=INVALID_QUANTITY
- B037 CHECK PRICE
 - B037_ADDITIONAL_TEXT=The displayed price may be the result of a keying error.
 - B037_HANDLING_CLASS_KEY=FatalErrorHandler
 - B037_ERROR_CODE=INVALID_ARGUMENT
 - B037_EXTENDED_ERROR_CODE=INVALID_PRICE
- B038 CHECK WEIGHT
 - B038_ADDITIONAL_TEXT=A weight of zero was entered for a weight item.
 - B038_HANDLING_CLASS_KEY=FatalErrorHandler
 - B038_ERROR_CODE=INVALID_ARGUMENT
 - B038_EXTENDED_ERROR_CODE=INVALID_WEIGHT
- B039 CHECK ITEM DATA
 - B039_ADDITIONAL_TEXT=The item record for the entered item contains invalid data. 1) The unit or deal price cannot be zero unless the item requires an entered price. 2) The department ID must be defined in Options->Department Groups personalization. 3) The miscellaneous account number must be defined in Options->Tender->Generic personalization. 4) Weight items and fuel items must use pricing method 0. 5) The linked item chain

- cannot exceed five items. 6) A fuel item cannot be used as an alias item code.
- B039_HANDLING_CLASS_KEY=FatalErrorHandler
- B039_ERROR_CODE=POS_APP_FAILURE
- B039_EXTENDED_ERROR_CODE=INVALID_APPLICATION_DATA
- B040 FILE ACCESS FAILED
 - B040_ADDITIONAL_TEXT=A required file access has failed.
 - B040_HANDLING_CLASS_KEY=ACEErrorHandlerB040
 - B040_ERROR_CODE=POS_APP_FAILURE
 - B040_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B041 CHECK ACCOUNT DATA
 - B041_ADDITIONAL_TEXT=This message is issued because one of the following conditions exists: 1) The tender verification record for the entered account number has a status code of 99 (reject always). 2) The tender verification record for the entered account number has a status code of 70 (user defined). 3) The tender verification record for the entered account number contains invalid status data.
 - B041_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B041_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B041_EXTENDED_ERROR_CODE=TENDER_NOT_AUTHORIZED
- B042 ITEM IS ON THE SCALE
 - B042_ADDITIONAL_TEXT=A weight cannot be keyed with an item on the scale.
 - B042_HANDLING_CLASS_KEY=FatalErrorHandler
 - B042_ERROR_CODE=SCALE_ERROR
 - B042_EXTENDED_ERROR_CODE=ITEM_IS_ON_THE_SCALE
 - B043 NO REFUND ON
- B043_ADDITIONAL_TEXT=The REFUND key has been pressed for an item code that cannot be refunded or for a department key that cannot be used with the REFUND key.
 - B043_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B043_ERROR_CODE=ITEM_NOT_RETURNABLE
 - B043_EXTENDED_ERROR_CODE=NONE
- B045 %1 DISCOUNT LIMIT CHECK
 - B045_ADDITIONAL_TEXT=A discount limit has been exceeded:
 - B045_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B045_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B045_EXTENDED_ERROR_CODE=DISCOUNT_LIMIT
- B046 ONE DISCOUNT PER TRANSACTION
 - B046_ADDITIONAL_TEXT=Only one discount may be taken during any transaction.
 - B046_HANDLING_CLASS_KEY=FatalErrorHandler
 - B046_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B046_EXTENDED_ERROR_CODE=DISCOUNT_LIMIT
- B047 CHECK DISCOUNT GROUP NUMBER
 - B047_ADDITIONAL_TEXT=The discount group number entered and displayed with the message is not defined.
 - B047_HANDLING_CLASS_KEY=FatalErrorHandler
 - B047_ERROR_CODE=INVALID_ARGUMENT
 - B047_EXTENDED_ERROR_CODE=INVALID_DISCOUNT_CODE
- B048 ACCOUNT NUMBER NEEDED
 - B048_ADDITIONAL_TEXT=An account number is needed with this tender for verification to be performed.
 - B048_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B048_ERROR_CODE=INVALID_ARGUMENT
 - B048_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_REQUIRED

- B049 INSERT DOCUMENT TO BE FRANKED
 - B049_ADDITIONAL_TEXT=A document is required in the document insert station for printing.
 - B049_HANDLING_CLASS_KEY=PrinterErrorHandler
 - B049_ERROR_CODE=NONE
 - B049_EXTENDED_ERROR_CODE=NONE
- B050 TENDER IS VERIFIED
 - B050_ADDITIONAL_TEXT=A tender entry submitted outside of a transaction for verification without cashing has been accepted.
 - B050_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B050_ERROR_CODE=NONE
 - B050_EXTENDED_ERROR_CODE=NONE
- B051 NOT ON FILE
 - B051_ADDITIONAL_TEXT=The account number entered and displayed in the message cannot be verified because it is not in the verification file.
 - B051_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B051_ERROR_CODE=INVALID_ARGUMENT
 - B051_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- B052 ACCOUNT LIMIT EXCEEDED (Quantity)
 - B052_ADDITIONAL_TEXT=Too many tenders have been made against this account number or in this transaction.
 - B052_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B052_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B052_EXTENDED_ERROR_CODE=ACCOUNT_TENDER_LIMIT
- B053 ACCOUNT LIMIT EXCEEDED (Amount)
 - B053_ADDITIONAL_TEXT=The tender amount entered and shown in the message exceeds the limit for this account number.
 - B053_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B053_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B053_EXTENDED_ERROR_CODE=ACCOUNT_TENDER_LIMIT
- B054 ACCOUNT LIMIT EXCEEDED (Total Amount)
 - B054_ADDITIONAL_TEXT=The total amount of tenders made against this account number exceeds the limit. The total for this account is shown in the message.
 - B054_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B054_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B054_EXTENDED_ERROR_CODE=ACCOUNT_TENDER_LIMIT
- B055 RISK 1
 - B055_ADDITIONAL_TEXT=The tender is rejected because the account is risk 1 as defined by your store procedures.
 - B055_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B055_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B055_EXTENDED_ERROR_CODE=RISK_1
- B056 RISK 2
 - B056_ADDITIONAL_TEXT=The tender is rejected because the account is risk 2 as defined by your store procedures.
 - B056_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B056_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B056_EXTENDED_ERROR_CODE=RISK_2
- B057 RISK 3
 - B057_ADDITIONAL_TEXT=The tender is rejected because the account is risk 3 as defined by your store procedures.
 - B057_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B057_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B057_EXTENDED_ERROR_CODE=RISK_3
- B058 RISK 4

- B058_ADDITIONAL_TEXT=The tender is rejected because the account is risk 4 as defined by your store procedures.
 - B058_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B058_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B058_EXTENDED_ERROR_CODE=RISK_4
- B059 CHECK OVERRIDE NUMBER
 - B059_ADDITIONAL_TEXT=The manager override number just entered is not defined.
 - B059_HANDLING_CLASS_KEY=FatalErrorHandler
 - B059_ERROR_CODE=INVALID_MANAGER_OVERRIDE_NUMBER
 - B059_EXTENDED_ERROR_CODE=NONE
- B060 CHANGE AMOUNT LIMIT CHECK
 - B060_ADDITIONAL_TEXT=The change limit for this tender type has been exceeded.
 - B060_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B060_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B060_EXTENDED_ERROR_CODE=CHANGE_AMOUNT_LIMIT
- B061 PRICE TOO LARGE
 - B061_ADDITIONAL_TEXT=The price of the item is larger than allowed.
 - B061_HANDLING_CLASS_KEY=FatalErrorHandler
 - B061_ERROR_CODE=MANAGER_OVERRIDE_REQUIRED
 - B061_EXTENDED_ERROR_CODE=NONE
- B062 KEYED PRICE LIMIT CHECK
 - B062_ADDITIONAL_TEXT=The price keyed for this item is too small for the department or too different from the price in the item record.
 - B062_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B062_ERROR_CODE=INVALID_ARGUMENT
 - B062_EXTENDED_ERROR_CODE=INVALID_PRICE
- B063 TENDER AMOUNT LIMIT CHECK
 - B063_ADDITIONAL_TEXT=The tender amount exceeds the limit for this tender type.
 - B063_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B063_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B063_EXTENDED_ERROR_CODE=TENDER_AMOUNT_LIMIT
- B064 COUPON MUST MATCH PREVIOUS SALE
 - B064_ADDITIONAL_TEXT=No item in the order matches the coupon, or coupons have already been taken for all of the matching items. If the department-level validation option is selected, this message might mean that the coupon item record has a different department number than the matching sale item record.
 - B064_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B064_ERROR_CODE=NO_ITEM_MATCH_FOR_COUPON
 - B064_EXTENDED_ERROR_CODE=NONE
- B065 COUPON VALUE EXCEEDS ITEM VALUE
 - B065_ADDITIONAL_TEXT=The coupon matches items sold in the transaction but exceeds the value of any of those items, perhaps because of coupon value multiplication.
 - B065_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B065_ERROR_CODE=COUPON_VALUE_EXCEEDS_ITEM_VALUE
 - B065_EXTENDED_ERROR_CODE=NONE
- B066 RETURN COUPON BEFORE ITEM CANCEL
 - B066_ADDITIONAL_TEXT=A coupon which has been tendered against the item being voided must be voided before the item.
 - B066_HANDLING_CLASS_KEY=FatalErrorHandler
 - B066_ERROR_CODE=RETURN_COUPON_BEFORE_ITEM_VOID
 - B066_EXTENDED_ERROR_CODE=NONE

- B067 CANCEL MUST MATCH PREVIOUS ENTRY
 - B067_ADDITIONAL_TEXT=A cancellation of an item, discount, or tender must match a previous entry to be accepted.
 - B067_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B067_ERROR_CODE=VOID_MUST_MATCH_PREVIOUS
 - B067_EXTENDED_ERROR_CODE=NONE
- B068 FULL KEYING SEQUENCE REQUIRED
 - B068_ADDITIONAL_TEXT=A VOID, ENTER or ENTER key sequence cannot be used to void or repeat the previous entry because it was a negative entry or it included a weight, quantity, or manager override.
 - B068_HANDLING_CLASS_KEY=FatalErrorHandler
 - B068_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B068_EXTENDED_ERROR_CODE=NONE
- B069 TENDERS/COUPONS MUST BE RETURNED
 - B069_ADDITIONAL_TEXT=A transaction has been voided after coupons or other tenders have been accepted.
 - B069_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B069_ERROR_CODE=NONE
 - B069_EXTENDED_ERROR_CODE=NONE
- B070 NEW PASSWORD ACCEPTED
 - B070_ADDITIONAL_TEXT=A new password has been entered at sign-on and has been accepted.
 - B070_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B070_ERROR_CODE=NONE
 - B070_EXTENDED_ERROR_CODE=NONE
- B071 PASSWORD MUST MATCH PREVIOUS ENTRY
 - B071_ADDITIONAL_TEXT=The new password entered at sign-on was not entered twice identically as is required for acceptance.
 - B071_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B071_ERROR_CODE=NONE
 - B071_EXTENDED_ERROR_CODE=NONE
- B072 PASSWORD NEEDED
 - B072_ADDITIONAL_TEXT=To exit the ****TERMINAL SECURED**** state, you must enter the password of the operator who secured the terminal.
 - B072_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B072_ERROR_CODE=INVALID_ARGUMENT
 - B072_EXTENDED_ERROR_CODE=PASSWORD_REQUIRED
- B073 NOT AVAILABLE OFFLINE
 - B073_ADDITIONAL_TEXT=When a terminal is offline, it must be re-initialized before it can be used for another sales transaction. A normal reload of the terminal or the running of diagnostic routines will suffice to re-initialize the terminal.
 - B073_HANDLING_CLASS_KEY=FatalErrorHandler
 - B073_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B073_EXTENDED_ERROR_CODE=NOT_ALLOWED_OFFLINE
- B074 TERMINAL HAS BEEN TRANSFERRED
 - B074_ADDITIONAL_TEXT=The terminal has been successfully transferred.
 - B074_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B074_ERROR_CODE=NONE
 - B074_EXTENDED_ERROR_CODE=NONE
- B075 CHECK PROCEDURE NUMBER
 - B075_ADDITIONAL_TEXT=The procedure number keyed is not defined.
 - B075_HANDLING_CLASS_KEY=FatalErrorHandler
 - B075_ERROR_CODE=INVALID_ARGUMENT
 - B075_EXTENDED_ERROR_CODE=INVALID_TRANSACTION_TYPE
- B076 AUTHORIZATION REQUIRED

- B076_ADDITIONAL_TEXT=The operator is not authorized for the attempted no-sale or non-sales procedure or is not authorized to perform a discount, refund, or miscellaneous transaction payout.
- B076_HANDLING_CLASS_KEY=FatalErrorHelper
- B076_ERROR_CODE=PROCEDURE_NOT_ALLOWED
- B076_EXTENDED_ERROR_CODE=NOT_AUTHORIZED
- B077 CHECK PASSWORD
 - B077_ADDITIONAL_TEXT=The password entered for a sign-on or special sign-on is incorrect.
 - B077_HANDLING_CLASS_KEY=FatalErrorHelper
 - B077_ERROR_CODE=INVALID_ARGUMENT
 - B077_EXTENDED_ERROR_CODE=INVALID_PASSWORD
- B078 CHECK OPERATOR NUMBER
 - B078_ADDITIONAL_TEXT=The operator number entered with a sign-on is invalid.
 - B078_HANDLING_CLASS_KEY=FatalErrorHelper
 - B078_ERROR_CODE=INVALID_ARGUMENT
 - B078_EXTENDED_ERROR_CODE=INVALID_ID
- B079 OPERATOR STILL ACTIVE ON %1
 - B079_ADDITIONAL_TEXT=An operator can sign on to only one terminal at a time. This operator is already signed on at the terminal number shown in the message.
 - B079_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B079_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B079_EXTENDED_ERROR_CODE=OPERATOR_ALREADY_ACTIVE
- B080 KEY OPERATOR,*,PASSWORD, SIGN-ON
 - B080_ADDITIONAL_TEXT=An invalid key sequence has been used on a terminal which is not yet signed on.
 - B080_HANDLING_CLASS_KEY=FatalErrorHelper
 - B080_ERROR_CODE=CONFIG_ERROR
 - B080_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B081 CLOSING PERIOD WAIT AND RETRY
 - B081_ADDITIONAL_TEXT=For a brief period when the close of a reporting period is selected at the manager's terminal, other terminals are prohibited from signing on or off. A terminal might also be prohibited from starting a new transaction.
 - B081_HANDLING_CLASS_KEY=FatalErrorHelper
 - B081_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B081_EXTENDED_ERROR_CODE=CLOSING_ACCOUNTING_PERIOD
- B082 PICKUP NEEDED SOON
 - B082_ADDITIONAL_TEXT=The pickup-needed-soon limit has been exceeded. Depending on the value of the Pickup-needed-soon at EOT option in Options->Security->Cash Drawer personalization, this message is displayed at either the beginning or the end of non-WIC transactions. The message is displayed at the specified time during the transactions until one of the following occurs: 1) A pickup is performed. 2) Message B083 is issued because the pickup-needed-now limit has also been exceeded.
 - B082_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B082_ERROR_CODE=NONE
 - B082_EXTENDED_ERROR_CODE=NONE
- B083 PICKUP REQUIRED NOW
 - B083_ADDITIONAL_TEXT=The pickup-needed-now limit has been exceeded. You cannot start a new transaction until enough cash has been picked up to bring this terminal below the pickup-needed-now limit.
 - B083_HANDLING_CLASS_KEY=FatalErrorHelper
 - B083_ERROR_CODE=PROCEDURE_NOT_ALLOWED

- B083_EXTENDED_ERROR_CODE=PICKUP_NEEDED
- B084 NOT AVAILABLE WHILE TRAINING
 - B084_ADDITIONAL_TEXT=A sign-on to tender listing, operator training, terminal monitor, or terminal transfer is prohibited when in training mode.
 - B084_HANDLING_CLASS_KEY=FatalErrorHandler
 - B084_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B084_EXTENDED_ERROR_CODE=NOT_ALLOWED_TRAINING
- B085 TILL EXCHANGE REQUIRED
 - B085_ADDITIONAL_TEXT=A till exchange is required as the first transaction of every new reporting period.
 - B085_HANDLING_CLASS_KEY=FatalErrorHandler
 - B085_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B085_EXTENDED_ERROR_CODE=TILL_EXCHANGE_NEEDED
 - B086 MONITOR ALREADY ACTIVE ON %1
- B086_ADDITIONAL_TEXT=The terminal monitor procedure may only be run at one terminal in the store at a time. This includes the manager's terminal. The procedure is already active at the terminal address shown in the message. If the terminal monitor procedure is active at the controller window, then the terminal number displayed is 999.
 - B086_HANDLING_CLASS_KEY=FatalErrorHandler
 - B086_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B086_EXTENDED_ERROR_CODE=MONITOR_ALREADY_ACTIVE
- B087 TERMINAL MUST MATCH PREVIOUS ENTRY
 - B087_ADDITIONAL_TEXT=The terminal number to transfer was not entered twice identically as is required for acceptance.
 - B087_HANDLING_CLASS_KEY=FatalErrorHandler
 - B087_ERROR_CODE=INVALID_ARGUMENT
 - B087_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- B088 CASH CHANGE OWED TO CUSTOMER
 - B088_ADDITIONAL_TEXT=A transaction containing an approved SurePOS ACE EPS tender has been voided or a tender exchange has been made to correct an invalid tender entry and, as a result, cash change is owed to the customer.
 - B088_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B088_ERROR_CODE=NONE
 - B088_EXTENDED_ERROR_CODE=NONE
- B089 CASH CHANGE NEEDED FROM CUSTOMER
 - B089_ADDITIONAL_TEXT=A tender exchange has been made to correct an invalid tender entry and, as a result, cash change which may previously have been given to the customer needs to be returned to the till.
 - B089_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B089_ERROR_CODE=NONE
 - B089_EXTENDED_ERROR_CODE=NONE
- B090 CHECK TENDER TYPE
 - B090_ADDITIONAL_TEXT=The tender key used is invalid for this function for one of these reasons: 1. Tender cashing cannot be used with cash and food stamps. 2. Tender listing is only valid for selected tender types. 3. Tender exchange is not allowed between selected tender types.
 - B090_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B090_ERROR_CODE=INVALID_ARGUMENT
 - B090_EXTENDED_ERROR_CODE=INVALID_TENDER_TYPE
- B092 INVALID FEE AMOUNT
 - B092_ADDITIONAL_TEXT=Only selected fee amounts can be entered with any tender type during a tender exchange procedure.
 - B092_HANDLING_CLASS_KEY=FatalErrorHandler
 - B092_ERROR_CODE=INVALID_ARGUMENT

- B092_EXTENDED_ERROR_CODE=INVALID_FEE_AMOUNT
- B094 PROCEDURE NOT AVAILABLE NOW
 - B094_ADDITIONAL_TEXT=A procedure has been attempted that is not available in the current transaction for one of the following reasons: 1. No-sale till exchange, tender verification, and till report are not allowed during a customer checkout transaction. 2. No help information is available in a customer checkout transaction. 3. A sign-off is not allowed within a checkout or non-sales transaction except for within a tender listing, terminal monitor, or terminal transfer transaction. 4. A sign-off is not allowed if the terminal experiences a failure in its write to totals retention. 5. The 100-SIGNON key sequence (Print the Pay Station OTR) and the 200-SIGNON key sequence (Refresh the Pay Station OTR) are only allowed on Pay Station terminals and are not allowed within a transaction.
 - B094_HANDLING_CLASS_KEY=FatalErrorHelper
 - B094_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B094_EXTENDED_ERROR_CODE=NONE
- B096 CHECK TERMINAL NUMBER
 - B096_ADDITIONAL_TEXT=The terminal number entered to be monitored or transferred is not active within the system.
 - B096_HANDLING_CLASS_KEY=FatalErrorHelper
 - B096_ERROR_CODE=INVALID_ARGUMENT
 - B096_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- B098 CHECK PIN PAD
 - B098_ADDITIONAL_TEXT=A personal identification number is needed for external tender processing and cannot be obtained from the MSR/PIN device.
 - B098_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B098_ERROR_CODE=PIN_PAD_ERROR
 - B098_EXTENDED_ERROR_CODE=PIN_COULD_NOT_BE_OBTAINED
- B099 INVALID REASON CODE %1
 - B099_ADDITIONAL_TEXT=The operator entered an undefined reason code for a returned item.
 - B099_HANDLING_CLASS_KEY=FatalErrorHelper
 - B099_ERROR_CODE=INVALID_ARGUMENT
 - B099_EXTENDED_ERROR_CODE=INVALID_REASON
- B100 RECURSIVE LINK CHAIN
 - B100_ADDITIONAL_TEXT=Two item records have links that point to each other. For example, item 1 is linked to item 2 and item 2 is linked to item 1. When item 1 is sold, item 2 is also sold. However, when the application detects that item 2 is linked to item 1 it reports this error.
 - B100_HANDLING_CLASS_KEY=FatalErrorHelper
 - B100_ERROR_CODE=POS_APP_FAILURE
 - B100_EXTENDED_ERROR_CODE=INVALID_APPLICATION_DATA
- B101 COUPON AS FIRST ITEM NOT ALLOWED
 - B101_ADDITIONAL_TEXT=You cannot scan a coupon before its corresponding item is sold because of a personalization default.
 - B101_HANDLING_CLASS_KEY=FatalErrorHelper
 - B101_ERROR_CODE=NO_ITEM_MATCH_FOR_COUPON
 - B101_EXTENDED_ERROR_CODE=NONE
- B103 ERROR WRITING TRANSACTION
 - B103_ADDITIONAL_TEXT=The terminal cannot access the transaction log. Data will be lost.
 - B103_HANDLING_CLASS_KEY=FatalErrorHelper
 - B103_ERROR_CODE=POS_APP_FAILURE
 - B103_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B104 EAS ITEM

- B104_ADDITIONAL_TEXT=An EAS item has been added to the transaction. This prompt is a reminder that the operator should deactivate the EAS device on the item.
- B104_HANDLING_CLASS_KEY=InformationalErrorHelper
- B104_ERROR_CODE=NONE
- B104_EXTENDED_ERROR_CODE=NONE
- B106 KEYED ENTRY NOT ALLOWED
 - B106_ADDITIONAL_TEXT=
 - B106_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B106_ERROR_CODE=NONE
 - B106_EXTENDED_ERROR_CODE=NONE
- B110 ONLY %1 DIGITS ALLOWED
 - B110_ADDITIONAL_TEXT=The operator entered more digits than the input data allowed. xx is the number of digits allowed for the input.
 - B110_HANDLING_CLASS_KEY=DefaultErrorHelper
 - B110_ERROR_CODE=INVALID_ARGUMENT
 - B110_EXTENDED_ERROR_CODE=DATA_OUT_OF_RANGE
- B111 INVALID PRINT REQUESTED
 - B111_ADDITIONAL_TEXT=A coding error has occurred, which led to the request of an invalid print message.
 - B111_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B111_ERROR_CODE=POS_APP_FAILURE
 - B111_EXTENDED_ERROR_CODE=UNRECOGNIZED_PRINT_CHARACTERS
- B112 REMOVE CHECK FROM PRINTER
 - B112_ADDITIONAL_TEXT=A check is in the Document Insert station.
 - B112_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B112_ERROR_CODE=NONE
 - B112_EXTENDED_ERROR_CODE=NONE
- B113 REMOVE FRANKED TENDER
 - B113_ADDITIONAL_TEXT=A print line needs to be printed on the customer receipt but cannot because a document is in the document insert station.
 - B113_HANDLING_CLASS_KEY=ACEErrorHelperB113
 - B113_ERROR_CODE=NONE
 - B113_EXTENDED_ERROR_CODE=NONE
- B117 PLEASE END TRANSACTION
 - B117_ADDITIONAL_TEXT=The transaction is approaching the upper limit on the number of items allowed in a transaction.
 - B117_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B117_ERROR_CODE=NONE
 - B117_EXTENDED_ERROR_CODE=NONE
- B118 NO MORE ITEMS ALLOWED
 - B118_ADDITIONAL_TEXT=The transaction is too large and must be ended. Only tender entries will be accepted at this time.
 - B118_HANDLING_CLASS_KEY=FatalErrorHelper
 - B118_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B118_EXTENDED_ERROR_CODE=MAX_NUMBER_OF_ITEMS
- B132 PREVIOUS ITEM STILL ON SCALE
 - B132_ADDITIONAL_TEXT=The scale still holds the item previously weighed.
 - B132_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B132_ERROR_CODE=SCALE_ERROR
 - B132_EXTENDED_ERROR_CODE=PREVIOUS_ITEM_ON_SCALE
- B133 PLEASE RE-WEIGH ITEM

- B133_ADDITIONAL_TEXT=The operator attempted to weigh an item while the following conditions existed: 1)The scale was not at zero during sign-on. 2)The scale has not returned to zero since sign-on.
 - B133_HANDLING_CLASS_KEY=FatalErrorHandler
 - B133_ERROR_CODE=SCALE_ERROR
 - B133_EXTENDED_ERROR_CODE=REWEIGH_ITEM
- B140 NO RECEIPT TO REPRINT
 - B140_ADDITIONAL_TEXT=A reprint receipt procedure was entered; a receipt is not available for reprint.
 - B140_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B140_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B140_EXTENDED_ERROR_CODE=NONE
- B141 INVALID DATE
 - B141_ADDITIONAL_TEXT=An incorrect date was entered.
 - B141_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B141_ERROR_CODE=INVALID_DATE
 - B141_EXTENDED_ERROR_CODE=NONE
- B142 NON-WIC ITEM
 - B142_ADDITIONAL_TEXT=The item is not identified in the item record file as a WIC item.
 - B142_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B142_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B142_EXTENDED_ERROR_CODE=NON_WIC_ITEM
- B143 CHECK FORMAT NOT UNDERSTOOD
 - B143_ADDITIONAL_TEXT=The account number information read from the check by the MICR device does not match a known format.
 - B143_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B143_ERROR_CODE=MICR_ERROR
 - B143_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_ERROR
- B144 THIS IS NOT A WIC TENDER
 - B144_ADDITIONAL_TEXT=A tender was selected that is not defined as WIC Tender
 - B144_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B144_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B144_EXTENDED_ERROR_CODE=NON_WIC_TENDER
- B145 WIC TENDER NOT ALLOWED
 - B145_ADDITIONAL_TEXT=WIC tender was selected for payment on a Non-WIC transaction.
 - B145_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B145_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B145_EXTENDED_ERROR_CODE=TENDER_TYPE_NOT_ALLOWED
- B146 TARE WEIGHT TOO LARGE
 - B146_ADDITIONAL_TEXT=The keyed tare weight exceeds specified limits.
 - B146_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B146_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B146_EXTENDED_ERROR_CODE=TARE_WEIGHT_TOO_LARGE
- B147 INVALID ID TYPE
 - B147_ADDITIONAL_TEXT=A SurePOS ACE EPS account ID was entered that is not defined in personalization.
 - B147_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B147_ERROR_CODE=INVALID_ARGUMENT
 - B147_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- B148 CHECK CONFIGURATION
 - B148_ADDITIONAL_TEXT=Either the application could not create a new balance inquiry transaction, or a valid ID was entered and an ID definition

- does not exist. Either the terminal does not have sufficient memory or a memory leak is occurring.
 - B148_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B148_ERROR_CODE=POS_APP_FAILURE
 - B148_EXTENDED_ERROR_CODE=NONE
- B149 INVALID ENTRY
 - B149_ADDITIONAL_TEXT=One of the following invalid entries was detected: 1. An EBT balance inquiry tender other than 1 or 2 was entered. 2. An invalid state code was entered. 3. A transit code more than 9 digits in length was entered. 4. The last 4 digits in the account number that was entered do not match the account number read from the MSR or entered on the keyboard.
 - B149_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B149_ERROR_CODE=INVALID_ARGUMENT
 - B149_EXTENDED_ERROR_CODE=NONE
- B150 VOUCHER EXPIRED
 - B150_ADDITIONAL_TEXT=The date on the voucher has expired. The customer must provide another valid tender.
 - B150_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B150_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B150_EXTENDED_ERROR_CODE=TENDER_EXPIRED
- B151 TENDER NOT LOANED
 - B151_ADDITIONAL_TEXT=A loan procedure was attempted and the loan was not processed.
 - B151_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B151_ERROR_CODE=POS_APP_FAILURE
 - B151_EXTENDED_ERROR_CODE=NONE
- B152 TENDER NOT PICKEDUP
 - B152_ADDITIONAL_TEXT=A pick up procedure was attempted and the pick up was not processed.
 - B152_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B152_ERROR_CODE=POS_APP_FAILURE
 - B152_EXTENDED_ERROR_CODE=NONE
- B169 TAKE FOREIGN TOTAL
 - B169_ADDITIONAL_TEXT=The operator attempted to tender foreign currency without taking a foreign currency total.
 - B169_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B169_ERROR_CODE=CONFIG_ERROR
 - B169_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B170 MISSING TENDER DETAILS
 - B170_ADDITIONAL_TEXT=The account number and card expiration date is missing.
 - B170_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B170_ERROR_CODE=INVALID_ARGUMENT
 - B170_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_REQUIRED
- B172 NEGATIVE BALANCE
 - B172_ADDITIONAL_TEXT=The transaction being processed has resulted in a negative balance.
 - B172_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B172_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B172_EXTENDED_ERROR_CODE=NEGATIVE_TRANSACTION_BALANCE
- B173 ITEM ENTERED TAKE TOTAL
 - B173_ADDITIONAL_TEXT=A subtotal is required.
 - B173_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B173_ERROR_CODE=CONFIG_ERROR

- B173_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B174 COUPON SCANNED TAKE TOTAL
 - B174_ADDITIONAL_TEXT=A subtotal is required.
 - B174_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B174_ERROR_CODE=CONFIG_ERROR
 - B174_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B175 KEYED ACCOUNT REQUIRES OVERRIDE
 - B175_ADDITIONAL_TEXT=A manager override is needed to accept the account number that was entered.
 - B175_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B175_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B175_EXTENDED_ERROR_CODE=NONE
- B176 TAX EXEMPT DISCGROUP NOT ALLOWED
 - B176_ADDITIONAL_TEXT=A discount was requested for an item that is not eligible for tax-exempt discounts.
 - B176_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B176_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B176_EXTENDED_ERROR_CODE=TAX_EXEMPTION_NOT_ALLOWED_FOR_ITEM
- B177 NOT DSCNTABLE
 - B177_ADDITIONAL_TEXT=A discount was requested for an item that is not eligible for a discount.
 - B177_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B177_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B177_EXTENDED_ERROR_CODE=DISCOUNT_NOT_ALLOWED_FOR_ITEM
- B178 ITEM DISCOUNTS NOT ALLOWED
 - B178_ADDITIONAL_TEXT=A discount was requested for an item that is not eligible for tax exempt discounts.
 - B178_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B178_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B178_EXTENDED_ERROR_CODE=ITEM_DISCOUNTS_NOT_ALLOWED
- B179 VERIFY ENTERED PRICE
 - B179_ADDITIONAL_TEXT=The operator is entering a price and using a lookup key for which Validate Price is turned on in personalization.
 - B179_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B179_ERROR_CODE=CONFIG_ERROR
 - B179_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B180 VOUCHER NOT YET VALID
 - B180_ADDITIONAL_TEXT=The voucher presented by the customer cannot be accepted. The date on the voucher is a date in the future. The customer must provide another form of tender.
 - B180_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B180_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B180_EXTENDED_ERROR_CODE=VOUCHER_NOT_YET_VALID
- B181 PLEASE CHECK VOUCHER DATES
 - B181_ADDITIONAL_TEXT=Check that the date on the voucher matches the date that was keyed.
 - B181_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B181_ERROR_CODE=NONE
 - B181_EXTENDED_ERROR_CODE=NONE
- B182 DATE MUST BE 8 DIGITS
 - B182_ADDITIONAL_TEXT=In response to the ENTER DATE OF BIRTH MMDDYYYY prompt, the operator entered a date that did not contain eight digits. The most common error is entering a date in the format MMDDYY.
 - B182_HANDLING_CLASS_KEY=FatalClearErrorHandler

- B182_ERROR_CODE=INVALID_ARGUMENT
 - B182_EXTENDED_ERROR_CODE=INVALID_DATE
- B183 VOID EBT FOODSTAMPS
 - B183_ADDITIONAL_TEXT=Foodstamps were tendered in an amount greater than the transaction total of foodstampable items. This might result from the customer overtendering or from the customer tendering and then voiding some foodstampable items from the transaction.
 - B183_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B183_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B183_EXTENDED_ERROR_CODE=FOODSTAMP_TENDER_IN_EXCESS_OF_FOODSTAMP_BALANCE
- B184 NO FUEL ON
 - B184_ADDITIONAL_TEXT=The VOLUME key was pressed for an item that is not a fuel item.
 - B184_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B184_ERROR_CODE=INVALID_ARGUMENT
 - B184_EXTENDED_ERROR_CODE=VOLUME_PROHIBITED
- B185 FUEL NEEDED
 - B185_ADDITIONAL_TEXT=A volume must be entered for a fuel item when it is sold.
 - B185_HANDLING_CLASS_KEY=ItemPromptErrorHandler
 - B185_ERROR_CODE=INVALID_ARGUMENT
 - B185_EXTENDED_ERROR_CODE=VOLUME_REQUIRED
- B186 CHECK FUEL
 - B186_ADDITIONAL_TEXT=A volume of zero was entered for a fuel item.
 - B186_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B186_ERROR_CODE=INVALID_ARGUMENT
 - B186_EXTENDED_ERROR_CODE=INVALID_VOLUME
- B187 CHECK EXTENDED PRICE
 - B187_ADDITIONAL_TEXT= The extended price that was entered is not valid.
 - B187_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B187_ERROR_CODE=INVALID_ARGUMENT
 - B187_EXTENDED_ERROR_CODE=EXTENDED_PRICE_TOO_LARGE
- B188 NO XPRICE ON
 - B188_ADDITIONAL_TEXT=The XPRICE key was pressed for an item that is not a fuel item.
 - B188_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B188_ERROR_CODE=INVALID_ARGUMENT
 - B188_EXTENDED_ERROR_CODE=EXTENDED_PRICE_PROHIBITED
- B300 TOO MANY EPS TENDERS
 - B300_ADDITIONAL_TEXT=The number of EPS tenders allowed has been exceeded for this transaction.
 - B300_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B300_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B300_EXTENDED_ERROR_CODE=NUMBER_OF_TENDERS_LIMIT
- B301 USE ANOTHER TENDER
 - B301_ADDITIONAL_TEXT=Accepting EPS tender has not been activated, or you are offline from the controller.
 - B301_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B301_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B301_EXTENDED_ERROR_CODE=CREDIT_NOT_AVAILABLE
- B302 PIN PAD NOT AVAILABLE
 - B302_ADDITIONAL_TEXT=The PIN pad is not operational at this time. The customer must use another form of tender.
 - B302_HANDLING_CLASS_KEY=FatalClearErrorHandler

- B302_ERROR_CODE=PIN_PAD_ERROR
 - B302_EXTENDED_ERROR_CODE=NONE
- B303 CHECK TENDER TYPE
 - B303_ADDITIONAL_TEXT=The selected tender type does not have a valid definition in personalization. The problem might be: The tender type is not defined. A PIN pad is being used and the tender type is not defined as an EPS tender. The tender is defined as an EPS tender, but the associated card plan ID is not valid.
 - B303_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B303_ERROR_CODE=INVALID_ARGUMENT
 - B303_EXTENDED_ERROR_CODE=INVALID_TENDER_TYPE
- B304 CUSTOMER MUST SIGN DOCUMENT
 - B304_ADDITIONAL_TEXT=
 - B304_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B304_ERROR_CODE=CUSTOMER_MUST_SIGN_DOCUMENT
 - B304_EXTENDED_ERROR_CODE=NONE
- B305 INVALID DATE
 - B305_ADDITIONAL_TEXT=The customer's debit or credit card contains an invalid expiration date, or an invalid expiration date was entered by the operator.
 - B305_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B305_ERROR_CODE=INVALID_ARGUMENT
 - B305_EXTENDED_ERROR_CODE=INVALID_EXPIRY_DATE
- B306 INVALID ACCOUNT NUMBER
 - B306_ADDITIONAL_TEXT=
 - B306_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B306_ERROR_CODE=INVALID_ARGUMENT
 - B306_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- B307 BALANCE INQUIRY FOR EBT ONLY
 - B307_ADDITIONAL_TEXT=
 - B307_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B307_ERROR_CODE=INVALID_ARGUMENT
 - B307_EXTENDED_ERROR_CODE=INVALID_TENDER_TYPE
- B308 CUSTOMER CHANGED AMOUNT
 - B308_ADDITIONAL_TEXT=transaction that is different from the total amount due. Either the amount exceeds or is less than the total amount due.
 - B308_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B308_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B308_EXTENDED_ERROR_CODE=CUSTOMER_ALTERED_AMOUNT_A
T_PINPAD
- B309 APPROVED
 - B309_ADDITIONAL_TEXT=The electronic tender provided by the customer is approved.
 - B309_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B309_ERROR_CODE=NONE
 - B309_EXTENDED_ERROR_CODE=NONE
- B310 REFUND NOT ALLOWED
 - B310_ADDITIONAL_TEXT=
 - B310_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B310_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B310_EXTENDED_ERROR_CODE=NONE
- B311 NOT AUTHORIZED
 - B311_ADDITIONAL_TEXT=The SurePOS ACE EPS tender was not approved. This message can occur if the host is not available and a local approval cannot be provided at your store.
 - B311_HANDLING_CLASS_KEY=OverrideErrorHelper

- B311_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B311_EXTENDED_ERROR_CODE=TENDER_NOT_AUTHORIZED
- B312 TOO LONG IN OFFLINE MODE
 - B312_ADDITIONAL_TEXT=The amount of time allowed for the store to process in stand-in mode has been exceeded.
 - B312_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B312_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B312_EXTENDED_ERROR_CODE=TOO_LONG_IN_STAND_IN
- B313 OFFLINE COUNT EXCEEDED
 - B313_ADDITIONAL_TEXT=The number of transactions allowed by the store to process in stand-in mode has been exceeded.
 - B313_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B313_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B313_EXTENDED_ERROR_CODE=STAND_IN_COUNT_LIMIT
- B314 OFFLINE AMOUNT EXCEEDED
 - B314_ADDITIONAL_TEXT=The total amount of money to be accepted in stand-in mode has been exceeded.
 - B314_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B314_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B314_EXTENDED_ERROR_CODE=STAND_IN_AMOUNT_LIMIT
- B315 OFFLINE NOT ALLOWED
 - B315_ADDITIONAL_TEXT=The servicer for this account does not allow the store to process EPS in stand-in mode.
 - B315_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B315_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B315_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_OFFLINE
- B316 RISK CODE 1
 - B316_ADDITIONAL_TEXT=The customer card is in the negative tender verification file and cannot be accepted. Follow store policy.
 - B316_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B316_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B316_EXTENDED_ERROR_CODE=RISK_1
- B317 FLOOR LIMIT 2 EXCEEDED
 - B317_ADDITIONAL_TEXT=The limit set by the store for the amount that can be processed without requiring a manager's approval.
 - B317_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B317_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B317_EXTENDED_ERROR_CODE=TENDER_FLOOR_LIMIT
- B318 CARD HAS EXPIRED
 - B318_ADDITIONAL_TEXT=The customer's card has expired.
 - B318_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B318_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B318_EXTENDED_ERROR_CODE=CARD_EXPIRED
- B319 CHECK EPS CONFIGURATION
 - B319_ADDITIONAL_TEXT=The servicer for this transaction is unknown to the host.
 - B319_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B319_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B319_EXTENDED_ERROR_CODE=UNKNOWN_SERVICER
- B320 INVALID DATA RETRY CARD SWIPE
 - B320_ADDITIONAL_TEXT=The data that was read from the value card cannot be processed. The data on the card might be corrupted or the card might have been swiped incorrectly.
 - B320_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B320_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B320_EXTENDED_ERROR_CODE=INVALID_CARD_DATA

- B321 CARD TYPE NOT KNOWN
 - B321_ADDITIONAL_TEXT=The customer's bank identification data is not in the BIN file.
 - B321_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B321_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B321_EXTENDED_ERROR_CODE=CARD_TYPE_UNKNOWN
- B322 FLOOR LIMIT 1 EXCEEDED
 - B322_ADDITIONAL_TEXT=The limit set by the store for the amount that can be processed without requiring a manager's approval.
 - B322_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B322_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B322_EXTENDED_ERROR_CODE=TENDER_FLOOR_LIMIT
- B323 CALL FOR AUTHORIZATION
 - B323_ADDITIONAL_TEXT=Before accepting the debit or credit payment presented by the customer, the approving agency must be contacted for an approval number.
 - B323_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B323_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B323_EXTENDED_ERROR_CODE=CALL_FOR_AUTHORIZATION
- B324 CANNOT COMPLETE
 - B324_ADDITIONAL_TEXT=The SurePOS ACE EPS transaction cannot complete at this time because the host is not available and the tender cannot be locally approved.
 - B324_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B324_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B324_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_OFFLINE
- B325 CUSTOMER CHOSE %1
 - B325_ADDITIONAL_TEXT=The customer chose a tender type at the PIN pad that is different from what the operator selected at the terminal. For example, the customer chose Debit and the operator chose Credit or another non-debit tender.
 - B325_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B325_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B325_EXTENDED_ERROR_CODE=CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD
- B326 CUSTOMER CANCELLED TENDER
 - B326_ADDITIONAL_TEXT=The customer swiped their debit, EBT, or Credit card and then decided to cancel.
 - B326_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B326_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B326_EXTENDED_ERROR_CODE=CUSTOMER_CANCELLED_TENDER_AT_PINPAD
- B327 OUT OF MEMORY GIVE CASH FOR TENDER
 - B327_ADDITIONAL_TEXT=While voiding a SurePOS ACE EPS transaction, a partial tender could not be voided due to a memory problem. The operator is instructed to give the customer cash for the tender which could not be voided.
 - B327_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B327_ERROR_CODE=POS_APP_FAILURE
 - B327_EXTENDED_ERROR_CODE=OUT_OF_MEMORY
- B328 VOID FAILED - GIVE CASH FOR TENDER
 - B328_ADDITIONAL_TEXT=The customer has paid with an EPS tender, such as a credit card. The transaction has already been processed by the bank.
 - B328_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B328_ERROR_CODE=PROCEDURE_NOT_ALLOWED

- B328_EXTENDED_ERROR_CODE=NONE
- B329 SECOND ID NEEDED
 - B329_ADDITIONAL_TEXT=The customer used a SurePOS ACE EPS tender in an amount for which your store requires two forms of identification, such as a driver's license, before accepting the tender.
 - B329_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B329_ERROR_CODE=NONE
 - B329_EXTENDED_ERROR_CODE=NONE
- B330 KEY ENTRY NOT ALLOWED
 - B330_ADDITIONAL_TEXT=This account number cannot be entered at the terminal keyboard. Depending on your store policies, the account number must be read from the customer's debit, EBT, or credit card through the PIN pad, MSR, or through a MICR device.
 - B330_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B330_ERROR_CODE=INVALID_ARGUMENT
 - B330_EXTENDED_ERROR_CODE=TRACK_DATA_REQUIRED
- B331 MSR NOT ALLOWED
 - B331_ADDITIONAL_TEXT=You attempted to enter an account number using a magnetic stripe reader other than the PIN pad reader. Depending on your store policies, the account number must only be read from the PIN pad or entered on the POS keyboard.
 - B331_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B331_ERROR_CODE=INVALID_ARGUMENT
 - B331_EXTENDED_ERROR_CODE=TRACK_DATA_PROHIBITED
- B332 MICR NOT ALLOWED
 - B332_ADDITIONAL_TEXT=You attempted to enter an account number using a MICR reader. Depending on your store policies, the account number must only be read from the PIN pad, MSR or manually entered using the keyboard.
 - B332_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B332_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B332_EXTENDED_ERROR_CODE=MICR_DATA_NOT_ALLOWED
- B333 NO EBT TENDERS CONFIGURED
 - B333_ADDITIONAL_TEXT=A customer attempted to use an Electronic Benefits Transfer (EBT) card. Depending on your store policies, the EBT tender may not be accepted.
 - B333_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B333_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B333_EXTENDED_ERROR_CODE=EBT_NOT_ALLOWED
- B334 VOUCHER REQUIRED
 - B334_ADDITIONAL_TEXT=The customer must present a voucher to complete this transaction.
 - B334_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B334_ERROR_CODE=ERROR_HANDLER_OTHER_ERROR
 - B334_EXTENDED_ERROR_CODE=NONE
- B335 UNABLE TO READ CARD
 - B335_ADDITIONAL_TEXT=
 - B335_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B335_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B335_EXTENDED_ERROR_CODE=INVALID_CARD_DATA
- B336 NUMBER IS NOT VALID
 - B336_ADDITIONAL_TEXT=The account number read from the customer's card has failed the modulo 10 check. The account number is not valid.
 - B336_HANDLING_CLASS_KEY=FatalClearErrorHandler
 - B336_ERROR_CODE=INVALID_ARGUMENT
 - B336_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER

- B337 SCANNED ENTRY NOT ALLOWED
 - B337_ADDITIONAL_TEXT=The operator attempted to use the scanner to enter information, such as an account number, and the scanner is not configured as a valid input device for that type of information.
 - B337_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B337_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B337_EXTENDED_ERROR_CODE=SCANNED_DATA_NOT_ALLOWED
- B338 INTERNAL EPS ERROR
 - B338_ADDITIONAL_TEXT=SurePOS ACE EPS has detected an error in the current tender message.
 - B338_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B338_ERROR_CODE=POS_APP_FAILURE
 - B338_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_ERROR
- B339 ENTER TO ACCEPT CLEAR TO REJECT
 - B339_ADDITIONAL_TEXT=The customer selected a different SurePOS ACE EPS payment type than the operator. Message B325 is displayed immediately before this message to indicate which tender the customer chose.
 - B339_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B339_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B339_EXTENDED_ERROR_CODE=CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD
- B340 PAYMENT TYPE MISMATCH
 - B340_ADDITIONAL_TEXT=The operator and customer selected different SurePOS ACE EPS tenders, one of which was EBT food stamps. Message B325 is displayed immediately before this message to indicate which tender the customer chose.
 - B340_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B340_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B340_EXTENDED_ERROR_CODE=CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD
- B341 WORKING KEY REQUIRED
 - B341_ADDITIONAL_TEXT=Personalization specifies that a store-level working key is required, but currently there is no valid working key available.
 - B341_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B341_ERROR_CODE=POS_APP_FAILURE
 - B341_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_ERROR
- B342 OPTIONS ERROR ID SPECIFIED
 - B342_ADDITIONAL_TEXT=A check ID has been specified for a non-check tender. Primary and secondary IDs are only valid with check tenders.
 - B342_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B342_ERROR_CODE=INVALID_ARGUMENT
 - B342_EXTENDED_ERROR_CODE=ID_PROHIBITED
- B343 OPTIONS ERROR INVALID ID DEVICE
 - B343_ADDITIONAL_TEXT=
 - B343_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B343_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B343_EXTENDED_ERROR_CODE=NONE
- B344 OPTIONS ERROR SECOND ID UNDEFINED
 - B344_ADDITIONAL_TEXT=One of the following conditions exists: 1) A check ID that has either Printer MICR or PIN pad MICR specified as an allowed input device in EPS->Check IDs->Input Devices has been specified as a secondary ID in EPS->EPS Tender->Check ID Flags. MICR devices can only be used for primary IDs. 2) A check ID that has either Keyboard MSR or PIN pad MSR specified as an allowed input device in EPS->Check

- IDs->Input Devices has been specified as a primary ID in EPS->EPS Tender->Check ID Flags. MSR devices can only be used for secondary IDs.
 - B344_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B344_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B344_EXTENDED_ERROR_CODE=NONE
- B345 OPTIONS ERROR ID TYPE NOT FOUND
 - B345_ADDITIONAL_TEXT=A check ID that has been selected in EPS->EPS Tender->Check ID Flags has not been defined in EPS->Check IDs.
 - B345_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B345_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B345_EXTENDED_ERROR_CODE=NONE
- B346 ERROR LOADING PIN PAD DEVICE
 - B346_ADDITIONAL_TEXT=An attempt was made to load either the PIN pad load file or the PIN pad parameters file into the PIN pad. The file could not be found.
 - B346_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B346_ERROR_CODE=PIN_PAD_ERROR
 - B346_EXTENDED_ERROR_CODE=LOAD_ERROR
- B347 PIN PAD NOT CAPABLE
 - B347_ADDITIONAL_TEXT=The specific PIN pad implementation does not support an optional SurePOS ACE EPS functionality. There is no alternate method of performing the operation, such as using the terminal MSR instead of the PIN pad MSR.
 - B347_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B347_ERROR_CODE=PIN_PAD_ERROR
 - B347_EXTENDED_ERROR_CODE=UNSUPPORTED_CAPABILITY
- B348 PIN PAD PARAMETERS ERROR
 - B348_ADDITIONAL_TEXT=The PIN pad parameters cannot be downloaded to the PIN pad due to an error in the PIN pad parameters file. The PIN pad will be unusable.
 - B348_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B348_ERROR_CODE=PIN_PAD_ERROR
 - B348_EXTENDED_ERROR_CODE=LOAD_ERROR
- B349 ZERO AMOUNT IS NOT ALLOWED
 - B349_ADDITIONAL_TEXT=A SurePOS ACE EPS tender was tendered with an amount of zero. After the zero amount is recognized, no further processing occurs.
 - B349_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B349_ERROR_CODE=INVALID_ARGUMENT
 - B349_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- B350 PIN PAD NOT ALLOWED
 - B350_ADDITIONAL_TEXT=An attempt was made to use the PIN pad to enter account data for a card plan that does not have the PIN pad configured as a valid input device.
 - B350_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B350_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B350_EXTENDED_ERROR_CODE=PIN_PAD_PROHIBITED
- B351 GET CUSTOMER OK
 - B351_ADDITIONAL_TEXT=
 - B351_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B351_ERROR_CODE=ERROR_HANDLER_OTHER_ERROR
 - B351_EXTENDED_ERROR_CODE=NONE
- B352 GIFT CARD AMT 0 COLLECT TO RECYCLE
 - B352_ADDITIONAL_TEXT=The customer has used a gift card as tender and the remaining balance on the gift card is now zero.
 - B352_HANDLING_CLASS_KEY=FatalClearErrorHelper

- B352_ERROR_CODE=NONE
 - B352_EXTENDED_ERROR_CODE=NONE
- B353 DUPLICATE VALUE CARD FOUND IN ORDER
 - B353_ADDITIONAL_TEXT=An attempt was made to sell a value card (phone card or gift card) but the account number that was entered matched the account number of a previous value card in the same order.
 - B353_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B353_ERROR_CODE=INVALID_ARGUMENT
 - B353_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- B354 VALUE CARD ITEM REMOVED FROM ORDER
 - B354_ADDITIONAL_TEXT=An attempt to activate or reload a value card failed, and the item has been removed from the order.
 - B354_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B354_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B354_EXTENDED_ERROR_CODE=NONE
- B355 ADDED TO ORDER %1
 - B355_ADDITIONAL_TEXT=The value card corresponding to account number xxxx has been successfully activated or reloaded.
 - B355_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B355_ERROR_CODE=NONE
 - B355_EXTENDED_ERROR_CODE=NONE
- B356 NO MATCHED ITEM VOID NOT ALLOWED
 - B356_ADDITIONAL_TEXT=An attempt was made to void a value card, but the specified account number did not match that of a value card previously sold in the order.
 - B356_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B356_ERROR_CODE=VOID_MUST_MATCH_PREVIOUS
 - B356_EXTENDED_ERROR_CODE=NONE
- B357 PREVIOUS HOST REQUEST PENDING
 - B357_ADDITIONAL_TEXT=An attempt was made to sell a value card or total the order while there was a pending activation request. A response must be received from the host or the request must time out before another value card can be sold or the order totaled.
 - B357_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B357_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B357_EXTENDED_ERROR_CODE=HOST_REQUEST_PENDING
- B358 VOID FROM ORDER %1
 - B358_ADDITIONAL_TEXT=The value card corresponding to account number xxxx has been voided.
 - B358_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B358_ERROR_CODE=NONE
 - B358_EXTENDED_ERROR_CODE=NONE
- B359 USE ANOTHER TENDER (GIPC DOWN)
 - B359_ADDITIONAL_TEXT=An EPS tender cannot be taken because the ACEGIPC task is not active on the controller.
 - B359_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B359_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B359_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_OFFLINE
- B360 VALUE CARDS MUST BE VOIDED
 - B360_ADDITIONAL_TEXT=The operator attempted to suspend a transaction that contains one or more value cards.
 - B360_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B360_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B360_EXTENDED_ERROR_CODE=VALUE_CARDS_MUST_BE_VOIDED
- B361 CARD REFUNDED %1

- B361_ADDITIONAL_TEXT=The value card corresponding to account number xxxx has been refunded.
 - B361_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B361_ERROR_CODE=NONE
 - B361_EXTENDED_ERROR_CODE=NONE
- B362 VALUE CARD ITEM VOID NOT APPROVED
 - B362_ADDITIONAL_TEXT=An attempt to void a value card item was not approved.
 - B362_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B362_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B362_EXTENDED_ERROR_CODE=NOT_APPROVED_BY_PAYMENT_SYSTEM
- B363 VALUE CARD ITEM REFUND NOT APPROVED
 - B363_ADDITIONAL_TEXT=An attempt to refund a value card item was not approved.
 - B363_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B363_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B363_EXTENDED_ERROR_CODE=NOT_APPROVED_BY_PAYMENT_SYSTEM
- B364 CARD RETURNED %1
 - B364_ADDITIONAL_TEXT=A 12-SignOn Return Transaction was attempted with a value card. The return of the value card corresponding to account number xxxx has been approved by the host.
 - B364_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B364_ERROR_CODE=NONE
 - B364_EXTENDED_ERROR_CODE=NONE
- B365 VALUE CARD ITEM RETURN NOT APPROVED
 - B365_ADDITIONAL_TEXT=A 12-SignOn Return Transaction was attempted with a value card. The return of the value card was not approved by the host.
 - B365_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B365_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B365_EXTENDED_ERROR_CODE=NOT_APPROVED_BY_PAYMENT_SYSTEM
- B366 CASHBACK GIVEN ONLY ON FINAL TENDER
 - B366_ADDITIONAL_TEXT=The customer specified a cash back amount for a partial tender that leaves a balance due for the transaction. Cash back is only allowed on tender for which the tendered amount equals the current balance due for the transaction.
 - B366_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B366_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B366_EXTENDED_ERROR_CODE=CASHBACK_ON_FINAL_TENDER_ONLY
- B367 USE VOID OF REF FOR THE VALUE CARD
 - B367_ADDITIONAL_TEXT=The operator refunded a value card, and then tried to sell the same value card.
 - B367_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B367_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B367_EXTENDED_ERROR_CODE=NONE
- B368 USE VOID FOR THIS VALUE CARD
 - B368_ADDITIONAL_TEXT=The operator sold a value card, and then tried to refund the same value card.
 - B368_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B368_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B368_EXTENDED_ERROR_CODE=NONE
- B369 SIGNATURE NOT AVAILABLE

- B369_ADDITIONAL_TEXT=The operator pressed CLEAR at the terminal when the 2x20 window displayed Wait For Signature. The operator then pressed ENTER to bypass the electronic signature. The store's personalization for this tender requires a manager's override if there is no electronic signature.
 - B369_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B369_ERROR_CODE=SIGNATURE_REQUIRED
 - B369_EXTENDED_ERROR_CODE=NONE
- B370 OVERRIDE NEEDED GIPC DOWN
 - B370_ADDITIONAL_TEXT=In a void transaction, one or more approved value cards cannot be voided because the ACEGIPC task is not active on the controller.
 - B370_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B370_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B370_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_OFFLINE
- B371 CANNOT PROCESS VALUE CARD-GIPC DOWN
 - B371_ADDITIONAL_TEXT=The operator attempted to sell, refund, void, or return a value card while hard totals is full and GIPC is down.
 - B371_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B371_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B371_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_OFFLINE
- B372 BALANCE INQUIRY NOT ENABLED
 - B372_ADDITIONAL_TEXT=The operator attempted to perform a balance inquiry. One of the following conditions exists: 1)The balance inquiry format for gift cards is set to zero (0). 2)The balance inquiry format that is specified for gift cards is not defined in Print->Balance Inquiry personalization.
 - B372_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B372_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B372_EXTENDED_ERROR_CODE=NONE
- B373 GIFT CARD NOT CONFIGURED
 - B373_ADDITIONAL_TEXT=The operator attempted to perform a balance inquiry. The gift card tender is not defined as a SurePOS ACE EPS tender.
 - B373_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B373_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B373_EXTENDED_ERROR_CODE=NONE
- B600 POINTS ENTRY REQUIRES CUSTOMER #
 - B600_ADDITIONAL_TEXT=When entering bonus points for a preferred customer, an account number must be entered.
 - B600_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B600_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B600_EXTENDED_ERROR_CODE=LOYALTY_NUMBER_REQUIRED
- B601 HOME STORE MUST REDEEM POINTS
 - B601_ADDITIONAL_TEXT=The customer can only redeem their bonus points at their home store (the store where their customer number was assigned).
 - B601_HANDLING_CLASS_KEY=FatalErrorHelper
 - B601_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B601_EXTENDED_ERROR_CODE=HOME_STORE_MUST_REDEEM_LOYALTY_POINTS
- B602 MORE POINTS NEEDED FOR DISCOUNT
 - B602_ADDITIONAL_TEXT=The customer needs additional bonus points before they are eligible for the associated discount.
 - B602_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B602_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B602_EXTENDED_ERROR_CODE=MORE_LOYALTY_POINTS_NEEDED
- B603 NO AVAILABLE DRAWER

- B603_ADDITIONAL_TEXT=The cash drawer is offline from the terminal or is missing.
 - B603_HANDLING_CLASS_KEY=FatalErrorHandler
 - B603_ERROR_CODE=CASH_DRAWER_ERROR
 - B603_EXTENDED_ERROR_CODE=NONE_AVAILABLE
- B604 CHECK KEYED CUSTOMER NUMBER
 - B604_ADDITIONAL_TEXT=The customer number that was entered is not correct.
 - B604_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B604_ERROR_CODE=INVALID_ARGUMENT
 - B604_EXTENDED_ERROR_CODE=INVALID_LOYALTY_NUMBER
- B605 CHECK CUSTOMER NUMBER %1
 - B605_ADDITIONAL_TEXT=The customer number xxx that was entered is not correct. xxx can be up to 18 digits.
 - B605_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B605_ERROR_CODE=INVALID_ARGUMENT
 - B605_EXTENDED_ERROR_CODE=INVALID_LOYALTY_NUMBER
- B606 REDEMPTION MUST BE POSITIVE
 - B606_ADDITIONAL_TEXT=The number of bonus points entered for redemption is more than the number of points the customer has accumulated.
 - B606_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B606_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B606_EXTENDED_ERROR_CODE=NONE
- B607 BONUS POINT LIMIT CHECK
 - B607_ADDITIONAL_TEXT=The number of bonus points included in this transaction exceeds the maximum number allowed for a single transaction.
 - B607_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B607_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B607_EXTENDED_ERROR_CODE=LOYALTY_POINTS_LIMIT
- B608 ASK FOR CARD CLEAR TO CONTINUE
 - B608_ADDITIONAL_TEXT=This is an informational prompt to remind the operator to ask for the preferred customer card.
 - B608_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B608_ERROR_CODE=NONE
 - B608_EXTENDED_ERROR_CODE=NONE
- B609 CHECK KEYED COUPON NUMBER
 - B609_ADDITIONAL_TEXT=The coupon number that was entered is not correct.
 - B609_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B609_ERROR_CODE=INVALID_ARGUMENT
 - B609_EXTENDED_ERROR_CODE=INVALID_ITEM_CODE
- B610 LIMITED NUMBER OF COUPONS PER ORDER
 - B610_ADDITIONAL_TEXT=The number of coupons entered is greater than the maximum allowed for one transaction.
 - B610_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B610_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B610_EXTENDED_ERROR_CODE=NUMBER_OF_COUPONS_LIMIT
- B611 COUPON HAS EXPIRED
 - B611_ADDITIONAL_TEXT=A coupon number was entered or scanned and the coupon is expired.
 - B611_HANDLING_CLASS_KEY=FatalErrorHandler
 - B611_ERROR_CODE=COUPON_EXPIRED
 - B611_EXTENDED_ERROR_CODE=NONE
- B612 MINIMUM SALE NOT SATISFIED

- B612_ADDITIONAL_TEXT=The customer does not have enough items to satisfy the minimum purchase requirement to allow acceptance of the coupon.
 - B612_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B612_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B612_EXTENDED_ERROR_CODE=MINIMUM_SALE_NOT_SATISFIED
- B613 PLEASE WRITE %1 ON COUPON
 - B613_ADDITIONAL_TEXT=Prompt to write the value xxx on an FSI coupon. xxx is a price that contains up to ten digits.
 - B613_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B613_ERROR_CODE=NONE
 - B613_EXTENDED_ERROR_CODE=NONE
- B614 ERROR ACCESSING CUSTOMER FILE
 - B614_ADDITIONAL_TEXT=The preferred customer account file cannot be accessed at this time.
 - B614_HANDLING_CLASS_KEY=FatalErrorHandler
 - B614_ERROR_CODE=POS_APP_FAILURE
 - B614_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B615 COUPON REQUIRES CUSTOMER NUMBER
 - B615_ADDITIONAL_TEXT=To use the coupon a preferred customer number must be entered.
 - B615_HANDLING_CLASS_KEY=FatalErrorHandler
 - B615_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B615_EXTENDED_ERROR_CODE=LOYALTY_NUMBER_REQUIRED
- B616 PREFERRED PROCESSING INACTIVE
 - B616_ADDITIONAL_TEXT=The preferred customer processing options are not defined in personalization.
 - B616_HANDLING_CLASS_KEY=FatalErrorHandler
 - B616_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B616_EXTENDED_ERROR_CODE=NONE
- B617 COUPON DOES NOT MATCH
 - B617_ADDITIONAL_TEXT=The coupon that the operator attempted to enter does not match any of the items that are currently in the transaction.
 - B617_HANDLING_CLASS_KEY=FatalErrorHandler
 - B617_ERROR_CODE=NO_ITEM_MATCH_FOR_COUPON
 - B617_EXTENDED_ERROR_CODE=NONE
- B618 DAILY CARD USES EXCEEDED LIMIT
 - B618_ADDITIONAL_TEXT=An attempt was made to use a Loyalty Program card more times in one day than the limit specified in Loyalty->General->Card Uses personalization.
 - B618_HANDLING_CLASS_KEY=FatalErrorHandler
 - B618_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B618_EXTENDED_ERROR_CODE=DAILY_LOYALTY_CARD_USAGE_LIMIT
- B619 CUSTOMER NOT AT REQUIRED LEVEL
 - B619_ADDITIONAL_TEXT=A preferred coupon with a non-zero coupon level failed to validate against the customer's status level.
 - B619_HANDLING_CLASS_KEY=FatalErrorHandler
 - B619_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B619_EXTENDED_ERROR_CODE=LOYALTY_COUPON_NOT_APPLICABLE_TO_CUSTOMER_STATUS_LEVEL
- B671 FUNCTION IS NOT AVAILABLE
 - B671_ADDITIONAL_TEXT=Suspend/Retrieve has been selected but store or terminal options have disabled the terminal from doing any suspension or retrieval.
 - B671_HANDLING_CLASS_KEY=FatalErrorHandler

- B671_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B671_EXTENDED_ERROR_CODE=NONE
- B672 RETRY WITH RETRIEVAL NUMBER
 - B672_ADDITIONAL_TEXT=The terminal number entered was incorrect.
 - B672_HANDLING_CLASS_KEY=FatalErrorHelper
 - B672_ERROR_CODE=INVALID_ARGUMENT
 - B672_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- B673 NO. OF SUSPENDS EXCEEDED
 - B673_ADDITIONAL_TEXT=Number of suspends has exceeded the maximum allowed.
 - B673_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B673_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B673_EXTENDED_ERROR_CODE=SUSPENDED_TRANSACTION_LIMIT
- B676 ACTIVE SUSPENDS CLEAR OR CONTINUE
 - B676_ADDITIONAL_TEXT=The operator or terminal has active suspends when signing off.
 - B676_HANDLING_CLASS_KEY=ACEErrorHelperB676
 - B676_ERROR_CODE=NONE
 - B676_EXTENDED_ERROR_CODE=NONE
- B677 RETRY WHERE SUSPENDED
 - B677_ADDITIONAL_TEXT=You attempted to retrieve a transaction on a terminal other than the one where the transaction was suspended.
 - B677_HANDLING_CLASS_KEY=FatalErrorHelper
 - B677_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B677_EXTENDED_ERROR_CODE=RETRIEVE_TRANSACTION_WHERE_SUSPENDED
- B678 UNABLE TO RETRIEVE DATA
 - B678_ADDITIONAL_TEXT=Transaction cannot be retrieved.
 - B678_HANDLING_CLASS_KEY=FatalErrorHelper
 - B678_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B678_EXTENDED_ERROR_CODE=UNABLE_TO_RETRIEVE_TRANSACTION
- B679 UNABLE TO UPDATE RETRIEVED RECORD
 - B679_ADDITIONAL_TEXT=The data from a suspended transaction cannot be updated.
 - B679_HANDLING_CLASS_KEY=FatalErrorHelper
 - B679_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B679_EXTENDED_ERROR_CODE=UNABLE_TO_RETRIEVE_TRANSACTION
- B680 TRANSACTION RETRIEVED ALREADY
 - B680_ADDITIONAL_TEXT=The operator attempted to retrieve a transaction that is no longer suspended.
 - B680_HANDLING_CLASS_KEY=FatalErrorHelper
 - B680_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B680_EXTENDED_ERROR_CODE=TRANSACTION_ALREADY_RETRIEVED
- B683 SAME OPERATOR MUST RETRIEVE
 - B683_ADDITIONAL_TEXT=The operator attempted to retrieve a transaction that can only be retrieved by the operator who suspended the transaction.
 - B683_HANDLING_CLASS_KEY=FatalErrorHelper
 - B683_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B683_EXTENDED_ERROR_CODE=SAME_OPERATOR_MUST_RETRIEVE_TRANSACTION
- B684 FORCE RETRIEVE REQUIRED
 - B684_ADDITIONAL_TEXT=An operator attempted to retrieve a suspended transaction that cannot be retrieved by this operator.

- B684_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B684_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B684_EXTENDED_ERROR_CODE=NONE
- B685 SUSPENSION INCOMPLETE
 - B685_ADDITIONAL_TEXT=An attempt to suspend a transaction did not successfully complete.
 - B685_HANDLING_CLASS_KEY=FatalErrorHelper
 - B685_ERROR_CODE=POS_APP_FAILURE
 - B685_EXTENDED_ERROR_CODE=NONE
- B686 RETRIEVAL INCOMPLETE
 - B686_ADDITIONAL_TEXT=An attempt to retrieve a transaction did not successfully complete.
 - B686_HANDLING_CLASS_KEY=FatalErrorHelper
 - B686_ERROR_CODE=POS_APP_FAILURE
 - B686_EXTENDED_ERROR_CODE=NONE
- B687 TRAINING MODE MISMATCH
 - B687_ADDITIONAL_TEXT=The operator attempted to recover a transaction in a training mode state that is different from the mode the transaction was in when it was suspended.
 - B687_HANDLING_CLASS_KEY=FatalErrorHelper
 - B687_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B687_EXTENDED_ERROR_CODE=NOT_ALLOWED_TRAINING
- B688 OVERRIDE NEEDED FOR SUSPEND
 - B688_ADDITIONAL_TEXT=A manager's override is required.
 - B688_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B688_ERROR_CODE=MANAGER_OVERRIDE_REQUIRED
 - B688_EXTENDED_ERROR_CODE=NONE
- B689 OVERRIDE NEEDED FOR RETRIEVE
 - B689_ADDITIONAL_TEXT=A manager's override is required.
 - B689_HANDLING_CLASS_KEY=FatalErrorHelper
 - B689_ERROR_CODE=MANAGER_OVERRIDE_REQUIRED
 - B689_EXTENDED_ERROR_CODE=NONE
- B690 %1 RESTRICTED ITEMS
 - B690_ADDITIONAL_TEXT=The self-checkout order that has just been retrieved contains xxx age-restricted items.
 - B690_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B690_ERROR_CODE=NONE
 - B690_EXTENDED_ERROR_CODE=NONE
- B691 VOID %1 ITEMS
 - B691_ADDITIONAL_TEXT=xxx items must be voided from the self-checkout originated order because the customer's age is not valid for those age-restricted items.
 - B691_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B691_ERROR_CODE=NONE
 - B691_EXTENDED_ERROR_CODE=NONE
- B864 CHECK FORMAT NOT UNDERSTOOD
 - B864_ADDITIONAL_TEXT=The account number information read from the check by the MICR device does not match a known format.
 - B864_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B864_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B864_EXTENDED_ERROR_CODE=MICR_DATA_NOT_ALLOWED
- B865 ERROR READING CHECK
 - B865_ADDITIONAL_TEXT=The MICR device could not read the check.
 - B865_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B865_ERROR_CODE=MICR_ERROR
 - B865_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_ERROR

- B870 VERIFY THE SIGNATURE ON CHECK
 - B870_ADDITIONAL_TEXT=The customer has tendered a check. Store policies require that the signature on the check be verified against a signature on an ID, such as a driver's license.
 - B870_HANDLING_CLASS_KEY=SignatureErrorHandler
 - B870_ERROR_CODE=VERIFY_SIGNATURE
 - B870_EXTENDED_ERROR_CODE=NONE
- B872 PRINTER COVER OPEN
 - B872_ADDITIONAL_TEXT=A printer cover is not closed.
 - B872_HANDLING_CLASS_KEY=PrinterErrorHandler
 - B872_ERROR_CODE=PRINTER_ERROR
 - B872_EXTENDED_ERROR_CODE=COVER_OPEN
- B873 INSERT CHECK TO BE PRINTED
 - B873_ADDITIONAL_TEXT=The personalization option for printing the face of checks has been selected. This is a prompt for the operator to insert the check into the printer.
 - B873_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B873_ERROR_CODE=NONE
 - B873_EXTENDED_ERROR_CODE=NONE
- B874 REMOVE CHECK FROM PRINTER
 - B874_ADDITIONAL_TEXT=The printer's processing of the check has been completed.
 - B874_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B874_ERROR_CODE=NONE
 - B874_EXTENDED_ERROR_CODE=NONE
- B876 PRESS DOCUMENT INSERT BUTTON
 - B876_ADDITIONAL_TEXT=Press the document insert button on the printer.
 - B876_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B876_ERROR_CODE=NONE
 - B876_EXTENDED_ERROR_CODE=NONE
- B877 CHANGE PASSWORD
 - B877_ADDITIONAL_TEXT=An operator attempted to sign on to a terminal and the operator password as expired. A new password must be entered before the operator is allowed to access the system.
 - B877_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B877_ERROR_CODE=PASSWORD_EXPIRED
 - B877_EXTENDED_ERROR_CODE=NONE
- B878 PASSWORD EXPIRES IN %1 DAYS
 - B878_ADDITIONAL_TEXT=Change password reminder prompt.
 - B878_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B878_ERROR_CODE=NONE
 - B878_EXTENDED_ERROR_CODE=NONE
- B879 PASSWORD NOT CHANGED
 - B879_ADDITIONAL_TEXT=An operator attempted to change their password and the system did not accept the new password. The new password is: 1) already being used 2) was previously used by this operator 3) does not conform to system password requirements
 - B879_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B879_ERROR_CODE=INVALID_ARGUMENT
 - B879_EXTENDED_ERROR_CODE=INVALID_PASSWORD
- B880 PRICE CHANGE NOT ALLOWED
 - B880_ADDITIONAL_TEXT=An operator attempted to change the price of an item that has the price change not allowed flag turned on in the item record.
 - B880_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B880_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B880_EXTENDED_ERROR_CODE=NONE

- B893 PRINTER OUT OF PAPER/PRINT COVER UP
 - B893_ADDITIONAL_TEXT=Printer is not available.
 - B893_HANDLING_CLASS_KEY=PrinterErrorHandler
 - B893_ERROR_CODE=PRINTER_ERROR
 - B893_EXTENDED_ERROR_CODE=NONE
- B894 INVALID PRINTER COMMAND
 - B894_ADDITIONAL_TEXT=The printer received an invalid command from the terminal.
 - B894_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B894_ERROR_CODE=POS_APP_FAILURE
 - B894_EXTENDED_ERROR_CODE=UNRECOGNIZED_PRINT_CHARACTERS
- B901 DEVICE OFFLINE
 - B901_ADDITIONAL_TEXT=One of the devices attached to the terminal is not communicating with the terminal.
 - B901_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B901_ERROR_CODE=POS_APP_FAILURE
 - B901_EXTENDED_ERROR_CODE=HARDWARE_PROBLEM
- B902 DISPLAY OFFLINE
 - B902_ADDITIONAL_TEXT=The operator or customer display is not communicating with the terminal.
 - B902_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B902_ERROR_CODE=DISPLAY_ERROR
 - B902_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B903 VIDEO DISPLAY OFFLINE
 - B903_ADDITIONAL_TEXT=The video display is not communicating with the terminal.
 - B903_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B903_ERROR_CODE=DISPLAY_ERROR
 - B903_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B904 SCALE OFFLINE
 - B904_ADDITIONAL_TEXT=The scale is not communicating with the terminal.
 - B904_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B904_ERROR_CODE=SCALE_ERROR
 - B904_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B905 KEYBOARD OFFLINE
 - B905_ADDITIONAL_TEXT=the keyboard is not communicating with the terminal.
 - B905_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B905_ERROR_CODE=KEYBOARD_ERROR
 - B905_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B906 TONE OFFLINE
 - B906_ADDITIONAL_TEXT=The tone device is not communicating with the terminal.
 - B906_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B906_ERROR_CODE=TONE_ERROR
 - B906_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B907 MSR OFFLINE
 - B907_ADDITIONAL_TEXT=The Magnetic Stripe Reader is not communicating with the terminal.
 - B907_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B907_ERROR_CODE=MSR_ERROR
 - B907_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B908 CASH DRAWER OFFLINE

- B908_ADDITIONAL_TEXT=The cash drawer is not communicating with the terminal.
 - B908_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B908_ERROR_CODE=CASH_DRAWER_ERROR
 - B908_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B909 UNKNOWN DEVICE OFFLINE
 - B909_ADDITIONAL_TEXT=An error occurred sending data to a coin dispenser that uses an RS-485 connection. As a result, no change was given.
 - B909_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B909_ERROR_CODE=COIN_DISPENSER_ERROR
 - B909_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B910 UNABLE TO OPEN COIN DISPENSER
 - B910_ADDITIONAL_TEXT=The coin dispenser was not initialized and will not dispense change.
 - B910_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B910_ERROR_CODE=COIN_DISPENSER_ERROR
 - B910_EXTENDED_ERROR_CODE=NONE
- B911 RESTRICTED QTY %2
 - B911_ADDITIONAL_TEXT=The sales transaction contains more items that are restricted by quantity than the maximum specified in Options.Restricted Sales personalization. If the extra items are not voided before the transaction is completed, an additional receipt will be printed. The message specifies the extra items by replacing xxx (which is on the second line of the 2x20 display) with one of the following: 1) The description of the restricted sale ID. This is the default version of the message. 2) The restricted sale ID, followed by the description of the restricted sale ID. 3) The word Group, followed by the restricted sale ID.
 - B911_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B911_ERROR_CODE=NONE
 - B911_EXTENDED_ERROR_CODE=NONE
- B999 PRESS CLEAR VALID TAX CODES ARE 1-4
 - B999_ADDITIONAL_TEXT=An incorrect number was entered for the tax plan.
 - B999_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B999_ERROR_CODE=INVALID_ARGUMENT
 - B999_EXTENDED_ERROR_CODE=NONE
- 11057 ENTER ACCOUNT NUMBER OR CLEAR
 - 11057_ADDITIONAL_TEXT=
 - 11057_HANDLING_CLASS_KEY=InformationalErrorHelper
 - 11057_ERROR_CODE=NONE
 - 11057_EXTENDED_ERROR_CODE=NONE
- 11076 ENTER DATE OF BIRTH MMDDYYYY
 - 11076_ADDITIONAL_TEXT=
 - 11076_HANDLING_CLASS_KEY=AgeRestrictedHelper
 - 11076_ERROR_CODE=AGE_RESTRICTED_ITEM
 - 11076_EXTENDED_ERROR_CODE=NONE
- 11033 ENTER PRICE OR PRESS CLEAR
 - 11033_ADDITIONAL_TEXT=
 - 11033_HANDLING_CLASS_KEY=ItemPromptErrorHelper
 - 11033_ERROR_CODE=INVALID_ARGUMENT
 - 11033_EXTENDED_ERROR_CODE=ITEM_PRICE_REQUIRED
- 11162 ENTER EXPIRATION DATE
 - 11162_ADDITIONAL_TEXT=An expiration date is required for this tender
 - 11162_HANDLING_CLASS_KEY=ExpirationDateErrorHelper
 - 11162_ERROR_CODE=INVALID_ARGUMENT

- 11162_EXTENDED_ERROR_CODE=EXPIRATION_DATE_REQUIRED
- 11516 ENTER VOID REASON CODE OR PRESS CLEAR
 - 11516_ADDITIONAL_TEXT=
 - 11516_HANDLING_CLASS_KEY=FatalErrorHandler
 - 11516_ERROR_CODE=INVALID_ARGUMENT
 - 11516_EXTENDED_ERROR_CODE=REASON_CODE_REQUIRED
- 13008 REMOVE CUSTOMER RECEIPT
 - 13008_ADDITIONAL_TEXT=The customer receipt needs to be removed
 - 13008_HANDLING_CLASS_KEY=InformationalErrorHandler
 - 13008_ERROR_CODE=NONE
 - 13008_EXTENDED_ERROR_CODE=NONE
- 13013 ENTER AUTHORIZATION CODE
 - 13013_ADDITIONAL_TEXT=An authorization code is needed to complete this tender
 - 13013_HANDLING_CLASS_KEY=AuthorizationCodeErrorHandler
 - 13013_ERROR_CODE=NONE
 - 13013_EXTENDED_ERROR_CODE=NONE
- 13031 ENTER VOUCHER NUMBER
 - 13031_ADDITIONAL_TEXT=A voucher number is needed to complete this tender
 - 13031_HANDLING_CLASS_KEY=VoucherNumberErrorHandler
 - 13031_ERROR_CODE=NONE
 - 13031_EXTENDED_ERROR_CODE=NONE
- 13052 ENTER CARD VALIDATION NUMBER
 - 13052_ADDITIONAL_TEXT=A card validation number is needed to complete this tender
 - 13052_HANDLING_CLASS_KEY=CardValidationNumberErrorHandler
 - 13052_ERROR_CODE=NONE
 - 13052_EXTENDED_ERROR_CODE=NONE

nextFile=usererror – This property is used to tell AEF the name of the next property file in the bundle chain.

Apperror.properties (GSA)

The apperror.properties file includes the following properties.

- Default error helper used when the key is not found in the error bundle or no helper class is specified.
 - APP_DEFAULT_ADDITIONAL_TEXT=The Default error helper was used because this error or its helper class is not defined in the error property files.
 - APP_DEFAULT_HANDLING_CLASS_KEY=AppDefaultErrorHandler
 - APP_DEFAULT_ERROR_CODE=CONFIG_ERROR
 - APP_DEFAULT_EXTENDED_ERROR_CODE=NO_ERROR_HELPER_CONFIGURED
- A001 Payment to Void not Found
 - A001_ADDITIONAL_TEXT=A request to void a payment has been made, but there is no payment in the transaction.
 - A001_HANDLING_CLASS_KEY=FatalErrorHandler
 - A001_ERROR_CODE=VOID_MUST_MATCH_PREVIOUS
 - A001_EXTENDED_ERROR_CODE=NONE
- A002 Rekey Entry
 - A002_ADDITIONAL_TEXT=The system is processing the previous entry. The key sequence is not accepted by the system.
 - A002_HANDLING_CLASS_KEY=FatalErrorHandler

- A002_ERROR_CODE=SYSTEM_BUSY
 - A002_EXTENDED_ERROR_CODE=NONE
- A003 Key Sequence is not valid
 - A003_ADDITIONAL_TEXT=There is an invalid key sequence.
 - A003_HANDLING_CLASS_KEY=FatalErrorHandler
 - A003_ERROR_CODE=CONFIG_ERROR
 - A003_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- A004 Key Sequence Does not Allow Data
 - A004_ADDITIONAL_TEXT=Data is not allowed with the function key pressed.
 - A004_HANDLING_CLASS_KEY=FatalErrorHandler
 - A004_ERROR_CODE=CONFIG_ERROR
 - A004_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- A005 Key Sequence Requires Data
 - A005_ADDITIONAL_TEXT=Data is required with the function key pressed.
 - A005_HANDLING_CLASS_KEY=FatalErrorHandler
 - A005_ERROR_CODE=CONFIG_ERROR
 - A005_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- A006 Number is not Valid
 - A006_ADDITIONAL_TEXT=The number of digits entered with the previous function key is incorrect or the numeric value exceeds the system's limits, or the numeric value does not fall within the proper range.
 - A006_HANDLING_CLASS_KEY=FatalErrorHandler
 - A006_ERROR_CODE=INVALID_ARGUMENT
 - A006_EXTENDED_ERROR_CODE=DATA_OUT_OF_RANGE
- A007 Check the Printer
 - A007_ADDITIONAL_TEXT=A printer error has occurred.
 - A007_HANDLING_CLASS_KEY=PrinterErrorHandler
 - A007_ERROR_CODE=PRINTER_ERROR
 - A007_EXTENDED_ERROR_CODE=NONE
- A008 Invalid Withdrawal Amount
 - A008_ADDITIONAL_TEXT=An invalid amount was entered.
 - A008_HANDLING_CLASS_KEY=FatalErrorHandler
 - A008_ERROR_CODE=INVALID_ARGUMENT
 - A008_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A009 Number of Items Exceeds Maximum
 - A009_ADDITIONAL_TEXT=The number of items selected in the store options (Data to Keep) has been exceeded.
 - A009_HANDLING_CLASS_KEY=FatalErrorHandler
 - A009_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - A009_EXTENDED_ERROR_CODE=MAX_NUMBER_OF_ITEMS
- A010 Keystrokes were Lost. Please Re-key
 - A010_ADDITIONAL_TEXT=The system cannot respond to the keystrokes. The system buffers some keystrokes, but the buffers were filled before accepting all keystrokes.
 - A010_HANDLING_CLASS_KEY=FatalErrorHandler
 - A010_ERROR_CODE=SYSTEM_BUSY
 - A010_EXTENDED_ERROR_CODE=NONE
- A011 Class Required Re-Key Item Sequence.
 - A011_ADDITIONAL_TEXT=The class number must be entered. Terminal options have been selected which require the class number to be entered for this department entry.
 - A011_HANDLING_CLASS_KEY=FatalErrorHandler
 - A011_ERROR_CODE=INVALID_ARGUMENT
 - A011_EXTENDED_ERROR_CODE=ITEM_CLASS_REQUIRED
- A012 Stock Required ReKey Item Sequence

- A012_ADDITIONAL_TEXT=The stock number must be entered. Terminal options have been selected which require stock number entry for this department, or the item record associated with the entered item code requires stock number entry.
 - A012_HANDLING_CLASS_KEY=FatalErrorHandler
 - A012_ERROR_CODE=INVALID_ARGUMENT
 - A012_EXTENDED_ERROR_CODE=ITEM_STOCK_REQUIRED
- A013 Dept is Prohibited
 - A013_ADDITIONAL_TEXT=The department number entered is prohibited. Terminal options department key control does not allow department number entry.
 - A013_HANDLING_CLASS_KEY=FatalErrorHandler
 - A013_ERROR_CODE=INVALID_ARGUMENT
 - A013_EXTENDED_ERROR_CODE=ITEM_DEPARTMENT_PROHIBITED
- A014 Class is Prohibited
 - A014_ADDITIONAL_TEXT=The class number is prohibited. Terminal options key control does not allow class number entry for this department.
 - A014_HANDLING_CLASS_KEY=FatalErrorHandler
 - A014_ERROR_CODE=INVALID_ARGUMENT
 - A014_EXTENDED_ERROR_CODE=ITEM_CLASS_PROHIBITED
- A015 Stock is Prohibited
 - A015_ADDITIONAL_TEXT=The stock number is prohibited. Terminal key control options do not allow stock number entry for this department.
 - A015_HANDLING_CLASS_KEY=FatalErrorHandler
 - A015_ERROR_CODE=INVALID_ARGUMENT
 - A015_EXTENDED_ERROR_CODE=ITEM_STOCK_PROHIBITED
- A016 Item Cannot be Returned
 - A016_ADDITIONAL_TEXT=The Item Record File indicates that this item is not returnable and that the item will not be accepted by the store for return.
 - A016_HANDLING_CLASS_KEY=FatalErrorHandler
 - A016_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A016_EXTENDED_ERROR_CODE=NONE
- A017 Extended Price is too Large
 - A017_ADDITIONAL_TEXT=The extended price is too large. The extended price calculated for this item has created an overflow condition and cannot be used by the application.
 - A017_HANDLING_CLASS_KEY=FatalErrorHandler
 - A017_ERROR_CODE=INVALID_ARGUMENT
 - A017_EXTENDED_ERROR_CODE=EXTENDED_PRICE_TOO_LARGE
- A018 Enter Item Before Price
 - A018_ADDITIONAL_TEXT=In the item record price change nonsales transaction, a price has been entered immediately following a completed price change. New prices that are entered must always be followed by the next item code. Item codes that are entered may be followed by a new price or a new item code. If you wish to change the price just entered, you must re-enter the item code.
 - A018_HANDLING_CLASS_KEY=FatalErrorHandler
 - A018_ERROR_CODE=CONFIG_ERROR
 - A018_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- A019 Price is Required
 - A019_ADDITIONAL_TEXT=The Item Record File indicates that a price is required for this item.
 - A019_HANDLING_CLASS_KEY=PriceRequiredErrorHandler
 - A019_ERROR_CODE=INVALID_ARGUMENT
 - A019_EXTENDED_ERROR_CODE=ITEM_PRICE_REQUIRED
- A020 No More Price Changes Allowed

- A020_ADDITIONAL_TEXT=The limit has been reached for the number of items allowed per transaction.
 - A020_HANDLING_CLASS_KEY=FatalErrorHandler
 - A020_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - A020_EXTENDED_ERROR_CODE=MAX_NUMBER_OF_PRICE_CHANGES
- A021 Cannot Change Pricing Methods 2/3
 - A021_ADDITIONAL_TEXT=The price of an item that uses pricing methods 2 or 3 cannot be changed at the terminal.
 - A021_HANDLING_CLASS_KEY=FatalErrorHandler
 - A021_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A021_EXTENDED_ERROR_CODE=NONE
- A022 Void Exceeds Counted Cash
 - A022_ADDITIONAL_TEXT=The void amount of the particular tender type entered exceeds the amount counted so far for that tender type in the terminal.
 - A022_HANDLING_CLASS_KEY=FatalErrorHandler
 - A022_ERROR_CODE=INVALID_ARGUMENT
 - A022_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A023 No Item to Void
 - A023_ADDITIONAL_TEXT=A VOID and ENTER keying sequence has been attempted after the previous entry has already been voided.
 - A023_HANDLING_CLASS_KEY=FatalErrorHandler
 - A023_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A023_EXTENDED_ERROR_CODE=NONE
- A024 Item has no Totals
 - A024_ADDITIONAL_TEXT=The item code entered during the nonsales obtain item movement transaction could not be found in the item movement file.
 - A024_HANDLING_CLASS_KEY=FatalErrorHandler
 - A024_ERROR_CODE=ITEM_NOT_FOUND
 - A024_EXTENDED_ERROR_CODE=NONE
- A025 Void Entry is Incorrect
 - A025_ADDITIONAL_TEXT=Line item void does not validate. Either the operator has attempted to void an item that has not been sold, or void a quantity greater than the number of items sold, or void an item using a different price than the original item entry.
 - A025_HANDLING_CLASS_KEY=FatalErrorHandler
 - A025_ERROR_CODE=VOID_MUST_MATCH_PREVIOUS
 - A025_EXTENDED_ERROR_CODE=NONE
- A026 Item not Found Re-try Item Entry
 - A026_ADDITIONAL_TEXT=The item entered is not on the Item Record File.
 - A026_HANDLING_CLASS_KEY=FatalErrorHandler
 - A026_ERROR_CODE=ITEM_NOT_FOUND
 - A026_EXTENDED_ERROR_CODE=NONE
- A027 Item is not Authorized for sale
 - A027_ADDITIONAL_TEXT=The line item entered is not authorized for sale. The item record for the item code entered indicates that the item cannot be sold.
 - A027_HANDLING_CLASS_KEY=FatalErrorHandler
 - A027_ERROR_CODE=ITEM_NOT_FOR_SALE
 - A027_EXTENDED_ERROR_CODE=NONE
- A028 No Data Found for Dept. nnnnnn.
 - A028_ADDITIONAL_TEXT=The department number (nnnnnn) entered or obtained from a list, was not found in the Department Totals or Item Record Files.

- A028_HANDLING_CLASS_KEY=FatalErrorHelper
 - A028_ERROR_CODE=INVALID_ARGUMENT
 - A028_EXTENDED_ERROR_CODE=INVALID_DEPARTMENT
- A029 Quantity is Required
 - A029_ADDITIONAL_TEXT=A quantity is required for the line item entered. The item record associated with the entered item code requires operator entry of a quantity in order to complete processing for the item code.
 - A029_HANDLING_CLASS_KEY=FatalErrorHelper
 - A029_ERROR_CODE=INVALID_ARGUMENT
 - A029_EXTENDED_ERROR_CODE=ITEM_QUANTITY_REQUIRED
- A030 Department Number not on File
 - A030_ADDITIONAL_TEXT=The department number entered is not defined in personalization.
 - A030_HANDLING_CLASS_KEY=FatalErrorHelper
 - A030_ERROR_CODE=INVALID_ARGUMENT
 - A030_EXTENDED_ERROR_CODE=INVALID_DEPARTMENT
- A031 D/C/S Entry is not Valid
 - A031_ADDITIONAL_TEXT=The D/C/S number is not valid as entered.
 - A031_HANDLING_CLASS_KEY=FatalErrorHelper
 - A031_ERROR_CODE=ITEM_NOT_FOUND
 - A031_EXTENDED_ERROR_CODE=NONE
- A033 SKU Entry is not Valid
 - A033_ADDITIONAL_TEXT=The SKU is not valid as entered. SKU validation was unsuccessful using the entered SKU number. Department SKU validation records can be contained in the terminal validation table. Both department SKU validation records and whole SKU validation records may reside in the Item Record File.
 - A033_HANDLING_CLASS_KEY=FatalErrorHelper
 - A033_ERROR_CODE=ITEM_NOT_FOUND
 - A033_EXTENDED_ERROR_CODE=NONE
- A034 Discount Percent is too Small
 - A034_ADDITIONAL_TEXT=The percentage of discount is less than the minimum selected in the terminal options.
 - A034_HANDLING_CLASS_KEY=FatalErrorHelper
 - A034_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - A034_EXTENDED_ERROR_CODE=DISCOUNT_PERCENTAGE_TOO_SMALL
- A035 Credit Plan not Defined
 - A035_ADDITIONAL_TEXT=The credit plan entered is not defined in personalization.
 - A035_HANDLING_CLASS_KEY=FatalErrorHelper
 - A035_ERROR_CODE=INVALID_ARGUMENT
 - A035_EXTENDED_ERROR_CODE=INVALID_CHARGE_PLAN
- A036 Deposit Amount is not Valid
 - A036_ADDITIONAL_TEXT=The deposit amount is either too high or too low. The deposit amount is either greater than the balance due or less than the minimum deposit amount selected in the terminal options.
 - A036_HANDLING_CLASS_KEY=FatalErrorHelper
 - A036_ERROR_CODE=INVALID_ARGUMENT
 - A036_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A037 Tender Amount is Required
 - A037_ADDITIONAL_TEXT=The tender amount must be entered. Terminal options require that a tender amount be entered with the CASH/TENDER key.
 - A037_HANDLING_CLASS_KEY=FatalErrorHelper
 - A037_ERROR_CODE=INVALID_ARGUMENT

- A037_EXTENDED_ERROR_CODE=AMOUNT_REQUIRED
- A038 Tender Type is not Allowed
 - A038_ADDITIONAL_TEXT=The tender type entered is not accepted at this terminal according to the terminal options file.
 - A038_HANDLING_CLASS_KEY=FatalErrorHelper
 - A038_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A038_EXTENDED_ERROR_CODE=TENDER_TYPE_NOT_ALLOWED
- A039 Fee Amount is not Valid
 - A039_ADDITIONAL_TEXT=The fee amount entered is too high or too low. The entered fee amount is either greater than the maximum fee amount or less than the minimum fee amount specified in the terminal options.
 - A039_HANDLING_CLASS_KEY=FatalErrorHelper
 - A039_ERROR_CODE=INVALID_ARGUMENT
 - A039_EXTENDED_ERROR_CODE=INVALID_FEE_AMOUNT
- A040 Payment has Already been Entered
 - A040_ADDITIONAL_TEXT=The payment has already been made within this transaction. Only one payment per transaction is allowed.
 - A040_HANDLING_CLASS_KEY=FatalErrorHelper
 - A040_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A040_EXTENDED_ERROR_CODE=PAYMENT_ALREADY_ENTERED
- A041 Payments are not Allowed
 - A041_ADDITIONAL_TEXT=The payment is not allowed. Terminal options indicate that payments are not allowed at this terminal.
 - A041_HANDLING_CLASS_KEY=FatalErrorHelper
 - A041_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A041_EXTENDED_ERROR_CODE=NONE
- A042 Returns are not Allowed
 - A042_ADDITIONAL_TEXT=The returns entered are not allowed. Terminal options indicate that returns are not allowed at this terminal, or the RETURN key is not allowed in the transaction.
 - A042_HANDLING_CLASS_KEY=FatalErrorHelper
 - A042_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A042_EXTENDED_ERROR_CODE=NONE
- A043 Allowances are not Allowed
 - A043_ADDITIONAL_TEXT=Allowances are not allowed. Terminal options indicate that allowances are not allowed at this terminal.
 - A043_HANDLING_CLASS_KEY=FatalErrorHelper
 - A043_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A043_EXTENDED_ERROR_CODE=NONE
- A044 Discount is not Allowed
 - A044_ADDITIONAL_TEXT=A discount has already been taken or a discount is not allowed.
 - A044_HANDLING_CLASS_KEY=FatalErrorHelper
 - A044_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A044_EXTENDED_ERROR_CODE=NONE
- A045 Discount Percent is too High
 - A045_ADDITIONAL_TEXT=The discount percentage exceeds the maximum discount limit specified in the terminal options.
 - A045_HANDLING_CLASS_KEY=FatalErrorHelper
 - A045_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - A045_EXTENDED_ERROR_CODE=DISCOUNT_PERCENTAGE_TOO_LARGE
- A046 Request not Allowed When Offline
 - A046_ADDITIONAL_TEXT=The operation requested is not allowed while offline. This message may be displayed if the terminal drops offline during a transaction in which access to files is necessary to complete the transaction.

- A046_HANDLING_CLASS_KEY=FatalErrorHelper
 - A046_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A046_EXTENDED_ERROR_CODE=NOT_ALLOWED_OFFLINE
- A047 Operation not Allowed in Re-Entry
 - A047_ADDITIONAL_TEXT=The operation requested is not allowed in re-entry.
 - A047_HANDLING_CLASS_KEY=FatalErrorHelper
 - A047_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A047_EXTENDED_ERROR_CODE=NOT_ALLOWED_REENTRY
- A048 Enter Line Item Before Allowance
 - A048_ADDITIONAL_TEXT=The line item must be entered before an allowance can be applied to an merchandise/nonmerchandise item entry.
 - A048_HANDLING_CLASS_KEY=FatalErrorHelper
 - A048_ERROR_CODE=INVALID_ARGUMENT
 - A048_EXTENDED_ERROR_CODE=ITEM_CODE_REQUIRED
- A049 Allowance Amount is too Large
 - A049_ADDITIONAL_TEXT=The allowance amount entered exceeds the extended price of the merchandise/nonmerchandise item entered prior to the allowance.
 - A049_HANDLING_CLASS_KEY=FatalErrorHelper
 - A049_ERROR_CODE=INVALID_ARGUMENT
 - A049_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A050 Department Key is not Valid
 - A050_ADDITIONAL_TEXT=The department key pressed is not a valid key. An automatic department key record does not exist for the department key pressed. The department key records are created during application personalization and reside in the terminal options.
 - A050_HANDLING_CLASS_KEY=FatalErrorHelper
 - A050_ERROR_CODE=INVALID_ARGUMENT
 - A050_EXTENDED_ERROR_CODE=INVALID_DEPARTMENT
- A052 Enter Line Item Before Discount
 - A052_ADDITIONAL_TEXT=Line item discount allowed only after line item or miscellaneous entries.
 - A052_HANDLING_CLASS_KEY=FatalErrorHelper
 - A052_ERROR_CODE=INVALID_ARGUMENT
 - A052_EXTENDED_ERROR_CODE=ITEM_CODE_REQUIRED
- A053 Zero Quantity is not Valid
 - A053_ADDITIONAL_TEXT=A zero quantity is not allowed during the cash count transaction.
 - A053_HANDLING_CLASS_KEY=FatalErrorHelper
 - A053_ERROR_CODE=INVALID_ARGUMENT
 - A053_EXTENDED_ERROR_CODE=INVALID_QUANTITY
- A054 Department Number is Required
 - A054_ADDITIONAL_TEXT=The terminal options key control table indicates that a department number is required for D/C/S entries.
 - A054_HANDLING_CLASS_KEY=FatalErrorHelper
 - A054_ERROR_CODE=INVALID_ARGUMENT
 - A054_EXTENDED_ERROR_CODE=ITEM_DEPARTMENT_REQUIRED
- A055 Tax Table is not Defined
 - A055_ADDITIONAL_TEXT=A requested tax table is not loaded in the terminal either because it is not indicated in the tax control file or because it was not found in its specified subdirectory during sales IPL when the terminal attempted to load it. This message can occur upon tax code entry or at the end of a transaction (if one or more items in the transaction require a tax table that has not been loaded).
 - A055_HANDLING_CLASS_KEY=FatalErrorHelper

- A055_ERROR_CODE=CONFIG_ERROR
 - A055_EXTENDED_ERROR_CODE=TAX_TABLE_NOT_FOUND
- A056 Enter Line Item Before Tax Code
 - A056_ADDITIONAL_TEXT=The line item must be entered before tax code.
 - A056_HANDLING_CLASS_KEY=FatalErrorHandler
 - A056_ERROR_CODE=INVALID_ARGUMENT
 - A056_EXTENDED_ERROR_CODE=ITEM_CODE_REQUIRED
- A057 Tax Code is not Defined
 - A057_ADDITIONAL_TEXT=The tax code requested could not be found in the tax code tables defined in terminal options. This message could also occur if the tax code was defined in the terminal options, but the position of the tax code within the tax code tables is greater than the number of tax tables indicated in the tax control file that was active during the last terminal load. In this case, message A140 would have been displayed at the terminal at least once, indicating a tax control file mismatch.
 - A057_HANDLING_CLASS_KEY=FatalErrorHandler
 - A057_ERROR_CODE=INVALID_ARGUMENT
 - A057_EXTENDED_ERROR_CODE=INVALID_TAX_CODE
- A058 Tax Code is Required
 - A058_ADDITIONAL_TEXT=The terminal options indicate that a tax code number must be entered.
 - A058_HANDLING_CLASS_KEY=FatalErrorHandler
 - A058_ERROR_CODE=INVALID_ARGUMENT
 - A058_EXTENDED_ERROR_CODE=TAX_CODE_REQUIRED
- A059 Charge Plan not Allowed
 - A059_ADDITIONAL_TEXT=The charge plan entered is not allowed. Valid charge plans are defined in personalization.
 - A059_HANDLING_CLASS_KEY=FatalErrorHandler
 - A059_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A059_EXTENDED_ERROR_CODE=CHARGE_PLAN_NOT_ALLOWED
- A060 Salesperson ID is Required
 - A060_ADDITIONAL_TEXT=A salesperson ID is required. Terminal options require the operator to enter a salesperson ID in order to sign on at the terminal.
 - A060_HANDLING_CLASS_KEY=FatalErrorHandler
 - A060_ERROR_CODE=INVALID_ARGUMENT
 - A060_EXTENDED_ERROR_CODE=ID_REQUIRED
- A061 Transaction Type is not Valid
 - A061_ADDITIONAL_TEXT=The transaction type entered is invalid. Valid IBM transaction types that may be selected are 0-9 followed by the ENTER key for Sales transactions or 0-15 followed by the NONSALES key for nonsales transactions.
 - A061_HANDLING_CLASS_KEY=FatalErrorHandler
 - A061_ERROR_CODE=INVALID_ARGUMENT
 - A061_EXTENDED_ERROR_CODE=INVALID_TRANSACTION_TYPE
- A062 Returns not Authorized
 - A062_ADDITIONAL_TEXT=Return transactions are not authorized.
 - A062_HANDLING_CLASS_KEY=FatalErrorHandler
 - A062_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A062_EXTENDED_ERROR_CODE=NOT_AUTHORIZED
- A063 Error Detected ERRN=xxxxxxx
 - A063_ADDITIONAL_TEXT=A terminal error has been sensed.
 - A063_HANDLING_CLASS_KEY=FatalErrorHandler
 - A063_ERROR_CODE=POS_APP_FAILURE
 - A063_EXTENDED_ERROR_CODE=NONE
- A064 Required File not Found

- A064_ADDITIONAL_TEXT=A required file was not found. nn is the file ID as follows: 01 Sequence Table (EALT@nnn), 02 Descriptor file (EALSDESC)03 Terminal Status file (EALTERMS), 04 Transaction Summary Log (EALTRANS), 05 Error Message file (EALAAAMF), 08 Store Options file (EALSOPTS), 09 Terminal Options file (EALT:nnn), 13 Sign-On Check file (EALSignO), 42 Tax Control file (EALC:nnnn).
 - A064_HANDLING_CLASS_KEY=FatalErrorHelper
 - A064_ERROR_CODE=POS_APP_FAILURE
 - A064_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A065 Return of Fee is not Allowed
 - A065_ADDITIONAL_TEXT=The terminal options indicate that the return of fees is not allowed.
 - A065_HANDLING_CLASS_KEY=FatalErrorHelper
 - A065_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A065_EXTENDED_ERROR_CODE=NONE
- A066 Return of Deposit not Allowed
 - A066_ADDITIONAL_TEXT=The terminal options indicate that the return of deposits is not allowed.
 - A066_HANDLING_CLASS_KEY=FatalErrorHelper
 - A066_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A066_EXTENDED_ERROR_CODE=NONE
- A067 Payment not Allowed in Cash Sale
 - A067_ADDITIONAL_TEXT=Payments cannot be made in a cash transaction. Payments are normally made in COD or layaway transactions.
 - A067_HANDLING_CLASS_KEY=FatalErrorHelper
 - A067_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A067_EXTENDED_ERROR_CODE=NONE
- A068 Refund is not Allowed
 - A068_ADDITIONAL_TEXT=The terminal options indicate that refunds are not allowed.
 - A068_HANDLING_CLASS_KEY=FatalErrorHelper
 - A068_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A068_EXTENDED_ERROR_CODE=NONE
- A069 Credit is not Allowed
 - A069_ADDITIONAL_TEXT=The terminal options indicate that credit transactions are not allowed.
 - A069_HANDLING_CLASS_KEY=FatalErrorHelper
 - A069_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A069_EXTENDED_ERROR_CODE=NONE
- A070 Transaction Cannot be Suspended
 - A070_ADDITIONAL_TEXT=The terminal options indicate that this transaction type cannot be suspended.
 - A070_HANDLING_CLASS_KEY=FatalErrorHelper
 - A070_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A070_EXTENDED_ERROR_CODE=NONE
- A071 Transaction Cannot be Retrieved
 - A071_ADDITIONAL_TEXT=Either a suspended transaction record that matches the transaction and terminal number entered could not be found, or there was an error reading the Suspended Transaction file.
 - A071_HANDLING_CLASS_KEY=FatalErrorHelper
 - A071_ERROR_CODE=INVALID_ARGUMENT
 - A071_EXTENDED_ERROR_CODE=NONE
- A072 Transaction Cannot be Voided
 - A072_ADDITIONAL_TEXT=The terminal options indicate that this transaction type cannot be voided, or a void previous transaction was attempted for: A transaction that does not exist in EALTLNDX, A transaction

type other than 1 through 4 or 6 through 9, An entered amount that does not agree with the original transaction amount, A transaction that has already been voided.

- A072_HANDLING_CLASS_KEY=FatalErrorHandler
- A072_ERROR_CODE=PROCEDURE_NOT_ALLOWED
- A072_EXTENDED_ERROR_CODE=NONE
- A073 Not an Operator Training Record.
 - A073_ADDITIONAL_TEXT=The operator is in operator training mode attempting to gain access to a layaway record that was not created while in training mode.
 - A073_HANDLING_CLASS_KEY=FatalErrorHandler
 - A073_ERROR_CODE=INVALID_ARGUMENT
 - A073_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- A074 Return Exceeds Departmental Limit
 - A074_ADDITIONAL_TEXT=The salesperson has attempted to return an item with a unit price that exceeds the return limit for that department as specified in the store options record.
 - A074_HANDLING_CLASS_KEY=OverrideErrorHandler
 - A074_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - A074_EXTENDED_ERROR_CODE=DEPARTMENT_RETURN_LIMIT
- A075 Transaction not Allowed in Void Mode
 - A075_ADDITIONAL_TEXT=The transaction selected is not valid while in Void Previous Line Item mode.
 - A075_HANDLING_CLASS_KEY=FatalErrorHandler
 - A075_ERROR_CODE=INVALID_ARGUMENT
 - A075_EXTENDED_ERROR_CODE=INVALID_TRANSACTION_TYPE
- A076 Salesperson is not Authorized
 - A076_ADDITIONAL_TEXT=The salesperson is not authorized for the requested activity. Authorization is contained in either the Authorization file or the store default authorization option.
 - A076_HANDLING_CLASS_KEY=OverrideErrorHandler
 - A076_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A076_EXTENDED_ERROR_CODE=NOT_AUTHORIZED
- A077 Password is Incorrect
 - A077_ADDITIONAL_TEXT=An incorrect password was entered. The password entered does not match the password in the Authorization file for the salesperson ID entered.
 - A077_HANDLING_CLASS_KEY=FatalErrorHandler
 - A077_ERROR_CODE=INVALID_ARGUMENT
 - A077_EXTENDED_ERROR_CODE=INVALID_PASSWORD
- A078 Operator Training Record
 - A078_ADDITIONAL_TEXT=The operator is in a normal sales layaway transaction attempting to gain access to a layaway record that was created during operator training mode.
 - A078_HANDLING_CLASS_KEY=FatalErrorHandler
 - A078_ERROR_CODE=INVALID_ARGUMENT
 - A078_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- A079 Operator Already Signed on
 - A079_ADDITIONAL_TEXT=The operator is signed on another terminal.
 - A079_HANDLING_CLASS_KEY=FatalErrorHandler
 - A079_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A079_EXTENDED_ERROR_CODE=OPERATOR_ALREADY_ACTIVE
- A080 Password is Required
 - A080_ADDITIONAL_TEXT=A password is required. The authorization record for the salesperson signing on or the default authorization store option indicates that a password must be entered.

- A080_HANDLING_CLASS_KEY=FatalErrorHelper
 - A080_ERROR_CODE=INVALID_ARGUMENT
 - A080_EXTENDED_ERROR_CODE=PASSWORD_REQUIRED
- A083 Amount is too Large
 - A083_ADDITIONAL_TEXT=In sales transactions, either the amount entered or the calculated discount amount is too large. In nonsales transactions, the total amount entered plus the accumulated total for loans, withdrawals, and counted cash cannot exceed 9 digits.
 - A083_HANDLING_CLASS_KEY=FatalErrorHelper
 - A083_ERROR_CODE=INVALID_ARGUMENT
 - A083_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A084 Not Allowed in Training Mode
 - A084_ADDITIONAL_TEXT=The operation requested cannot be entered in training mode.
 - A084_HANDLING_CLASS_KEY=FatalErrorHelper
 - A084_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A084_EXTENDED_ERROR_CODE=NOT_ALLOWED_TRAINING
- A085 For Help, Press Non-sales and Enter
 - A085_ADDITIONAL_TEXT=Nonsales help is selected by pressing NONSALES and ENTER.
 - A085_HANDLING_CLASS_KEY=FatalErrorHelper
 - A085_ERROR_CODE=INVALID_ARGUMENT
 - A085_EXTENDED_ERROR_CODE=INVALID_TRANSACTION_TYPE
- A086 Request is not Valid
 - A086_ADDITIONAL_TEXT=The selection requested is invalid. This message is displayed during the total readout/reset transaction when a 1, 2, or 9 must be entered to request reset/close, reset only, or end, respectively. If something other than these three numbers is entered, the error occurs. It is also displayed when a VAT transaction is requested and all VAT tax percentages are zero.
 - A086_HANDLING_CLASS_KEY=FatalErrorHelper
 - A086_ERROR_CODE=INVALID_ARGUMENT
 - A086_EXTENDED_ERROR_CODE=NONE
- A087 Transaction Type is Required
 - A087_ADDITIONAL_TEXT=The transaction type must be entered. A default transaction type has not been defined in the terminal options.
 - A087_HANDLING_CLASS_KEY=FatalErrorHelper
 - A087_ERROR_CODE=INVALID_ARGUMENT
 - A087_EXTENDED_ERROR_CODE=TRANSACTION_TYPE_REQUIRED
- A088 Sends are not allowed
 - A088_ADDITIONAL_TEXT=The terminal options indicate that sends are not allowed.
 - A088_HANDLING_CLASS_KEY=FatalErrorHelper
 - A088_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A088_EXTENDED_ERROR_CODE=NONE
- A089 Transaction Type is not Allowed
 - A089_ADDITIONAL_TEXT=The terminal options indicate that the transaction type selected is not allowed to be performed at this terminal.
 - A089_HANDLING_CLASS_KEY=FatalErrorHelper
 - A089_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A089_EXTENDED_ERROR_CODE=NONE
- A090 Transaction not Allowed when Offline
 - A090_ADDITIONAL_TEXT=The transaction type selected is not allowed while the terminal is in offline mode.
 - A090_HANDLING_CLASS_KEY=FatalErrorHelper
 - A090_ERROR_CODE=PROCEDURE_NOT_ALLOWED

- A090_EXTENDED_ERROR_CODE=NOT_ALLOWED_OFFLINE
- A091 Transaction not Allowed in Re-entry
 - A091_ADDITIONAL_TEXT=The transaction type selected is not allowed while the terminal is in re-entry mode.
 - A091_HANDLING_CLASS_KEY=FatalErrorHandler
 - A091_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A091_EXTENDED_ERROR_CODE=NOT_ALLOWED_REENTRY
- A092 Transaction not Allowed in Training
 - A092_ADDITIONAL_TEXT=The transaction type selected is not allowed while the terminal is in operator training mode.
 - A092_HANDLING_CLASS_KEY=FatalErrorHandler
 - A092_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A092_EXTENDED_ERROR_CODE=NOT_ALLOWED_TRAINING
- A093 Suspends are not Allowed
 - A093_ADDITIONAL_TEXT=Explanation: Either the suspension of transaction is not allowed in re-entry mode or the suspension of transactions is not allowed at this terminal. The latter restriction is contained in the terminal options.
 - A093_HANDLING_CLASS_KEY=FatalErrorHandler
 - A093_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A093_EXTENDED_ERROR_CODE=NONE
- A095 Authorization Code is not Valid
 - A095_ADDITIONAL_TEXT=The authorization code entered is invalid. An authorization code is required for credit authorization.
 - A095_HANDLING_CLASS_KEY=FatalErrorHandler
 - A095_ERROR_CODE=INVALID_ARGUMENT
 - A095_EXTENDED_ERROR_CODE=INVALID_AUTHORIZATION_CODE
- A096 Layaway Account Already Exists
 - A096_ADDITIONAL_TEXT=The layaway account number entered is already on file.
 - A096_HANDLING_CLASS_KEY=FatalErrorHandler
 - A096_ERROR_CODE=INVALID_ARGUMENT
 - A096_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- A097 Account Number is Required
 - A097_ADDITIONAL_TEXT=The layaway account number must be entered for this type of layaway transaction.
 - A097_HANDLING_CLASS_KEY=FatalErrorHandler
 - A097_ERROR_CODE=INVALID_ARGUMENT
 - A097_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_REQUIRED
- A098 Account Number was not Found
 - A098_ADDITIONAL_TEXT=The layaway account number entered is not on file.
 - A098_HANDLING_CLASS_KEY=FatalErrorHandler
 - A098_ERROR_CODE=INVALID_ARGUMENT
 - A098_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- A099 Payment is not Valid. Bal Due is 0.
 - A099_ADDITIONAL_TEXT=The layaway payment entered cannot be accepted. All payments to this account have been received. Another payment cannot be made since this account is paid in full.
 - A099_HANDLING_CLASS_KEY=FatalErrorHandler
 - A099_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A099_EXTENDED_ERROR_CODE=ACCOUNT_PAID_IN_FULL
- A100 Layaway Payment Exceeds Balance Due.
 - A100_ADDITIONAL_TEXT=The layaway payment entered exceeds the balance due.
 - A100_HANDLING_CLASS_KEY=FatalErrorHandler

- A100_ERROR_CODE=INVALID_ARGUMENT
 - A100_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A101 Item is not on Layaway
 - A101_ADDITIONAL_TEXT=The item keyed is not on layaway for this account.
 - A101_HANDLING_CLASS_KEY=FatalErrorHandler
 - A101_ERROR_CODE=ITEM_NOT_FOUND
 - A101_EXTENDED_ERROR_CODE=NONE
- A102 No Cancelled Layaway Item to Void
 - A102_ADDITIONAL_TEXT=The item keyed has not been cancelled.
 - A102_HANDLING_CLASS_KEY=FatalErrorHandler
 - A102_ERROR_CODE=INVALID_ARGUMENT
 - A102_EXTENDED_ERROR_CODE=INVALID_ITEM_CODE
- A103 Void Exceeds Cancelled Quantity
 - A103_ADDITIONAL_TEXT=The layaway void quantity is greater than the cancel quantity.
 - A103_HANDLING_CLASS_KEY=FatalErrorHandler
 - A103_ERROR_CODE=INVALID_ARGUMENT
 - A103_EXTENDED_ERROR_CODE=INVALID_QUANTITY
- A104 Layaway Cancel Quantity not Valid
 - A104_ADDITIONAL_TEXT=The layaway cancel quantity is invalid.
 - A104_HANDLING_CLASS_KEY=FatalErrorHandler
 - A104_ERROR_CODE=INVALID_ARGUMENT
 - A104_EXTENDED_ERROR_CODE=INVALID_QUANTITY
- A105 Last Payment Must Equal Bal Due
 - A105_ADDITIONAL_TEXT=The last payment or twentieth payment made to a layaway account must equal the balance due.
 - A105_HANDLING_CLASS_KEY=FatalErrorHandler
 - A105_ERROR_CODE=INVALID_ARGUMENT
 - A105_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A106 No Listing Data was Found
 - A106_ADDITIONAL_TEXT=The current period tender listing file selected is not on the file directory.
 - A106_HANDLING_CLASS_KEY=FatalErrorHandler
 - A106_ERROR_CODE=POS_APP_FAILURE
 - A106_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A107 Dept. Total List File not Found
 - A107_ADDITIONAL_TEXT=The department totals list file does not exist.
 - A107_HANDLING_CLASS_KEY=FatalErrorHandler
 - A107_ERROR_CODE=POS_APP_FAILURE
 - A107_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A108 List Number is not Defined
 - A108_ADDITIONAL_TEXT=The department totals list number entered during the obtain department totals transaction could not be found in the Department Totals List file.
 - A108_HANDLING_CLASS_KEY=FatalErrorHandler
 - A108_ERROR_CODE=INVALID_ARGUMENT
 - A108_EXTENDED_ERROR_CODE=INVALID_DEPARTMENT_TOTALS_LIST_NUMBER
- A109 Message nnn Missing or Deleted
 - A109_ADDITIONAL_TEXT=The message number nnn was deleted during personalization.
 - A109_HANDLING_CLASS_KEY=FatalErrorHandler
 - A109_ERROR_CODE=CONFIG_ERROR
 - A109_EXTENDED_ERROR_CODE=MISSING_DESCRIPTOR

- A112 Item Data Must be Wanded. This error should never be generated by calling the automation API.
 - A112_ADDITIONAL_TEXT=An attempt was made to key in item data immediately after a price change ticket was wanded. An item ticket must be wanded immediately after a price change ticket.
 - A112_HANDLING_CLASS_KEY=FatalErrorHandler
 - A112_ERROR_CODE=NONE
 - A112_EXTENDED_ERROR_CODE=NONE
 - A113 Batch Number is not Valid
- A113_ADDITIONAL_TEXT=The batch number for the delayed price change must be between 10101 and 999996 inclusive.
 - A113_HANDLING_CLASS_KEY=FatalErrorHandler
 - A113_ERROR_CODE=INVALID_ARGUMENT
 - A113_EXTENDED_ERROR_CODE=INVALID_BATCH_NUMBER
- A114 Immediate Price Change not Allowed
 - A114_ADDITIONAL_TEXT=The operator is not authorized for the immediate price change.
 - A114_HANDLING_CLASS_KEY=FatalErrorHandler
 - A114_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A114_EXTENDED_ERROR_CODE=NONE
- A115 Delayed Price Change not Allowed
 - A115_ADDITIONAL_TEXT=The operator is not authorized for the delayed price change.
 - A115_HANDLING_CLASS_KEY=FatalErrorHandler
 - A115_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A115_EXTENDED_ERROR_CODE=NONE
- A116 Term. Monitor File Failure
 - A116_ADDITIONAL_TEXT=There has been a system error while accessing the terminal monitor file.
 - A116_HANDLING_CLASS_KEY=FatalErrorHandler
 - A116_ERROR_CODE=POS_APP_FAILURE
 - A116_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A117 Terminal Number not Found
 - A117_ADDITIONAL_TEXT=The terminal number entered during the nonsales terminal monitor transaction could not be found in the terminal status file.
 - A117_HANDLING_CLASS_KEY=FatalErrorHandler
 - A117_ERROR_CODE=INVALID_ARGUMENT
 - A117_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- A118 Terminal Cannot Monitor Itself
 - A118_ADDITIONAL_TEXT=A terminal cannot monitor itself.
 - A118_HANDLING_CLASS_KEY=FatalErrorHandler
 - A118_ERROR_CODE=INVALID_ARGUMENT
 - A118_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- A119 Check MSR/PIN pad
 - A119_ADDITIONAL_TEXT=A communication error has occurred between the MSR/PIN pad and the terminal.
 - A119_HANDLING_CLASS_KEY=FatalErrorHandler
 - A119_ERROR_CODE=PIN_PAD_ERROR
 - A119_EXTENDED_ERROR_CODE=COMMUNICATION_ERROR
- A120 Payment Type Mismatch
 - A120_ADDITIONAL_TEXT=The customer has selected a payment type on the MSR/PIN pad which does not match the tender type entered at the terminal.
 - A120_HANDLING_CLASS_KEY=FatalErrorHandler
 - A120_ERROR_CODE=TENDER_NOT_ACCEPTED

- A120_EXTENDED_ERROR_CODE=CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD
- A121 Tender Amount has been Altered
 - A121_ADDITIONAL_TEXT=The customer has altered the tender amount at the MSR/PIN pad so that the amount no longer matches the balance due or the amount entered by the operator.
 - A121_HANDLING_CLASS_KEY=FatalErrorHelper
 - A121_ERROR_CODE=TENDER_NOT_ACCEPTED
 - A121_EXTENDED_ERROR_CODE=CUSTOMER_ALTERED_AMOUNT_AT_PINPAD
- A122 Verification Timeout
 - A122_ADDITIONAL_TEXT=The terminal has not received a response for an external tender verification request within the selected time.
 - A122_HANDLING_CLASS_KEY=FatalErrorHelper
 - A122_ERROR_CODE=TENDER_NOT_ACCEPTED
 - A122_EXTENDED_ERROR_CODE=VERIFICATION_TIMEOUT
- A124 Change Over Limit
 - A124_ADDITIONAL_TEXT=The change limit of the debit tender has been exceeded.
 - A124_HANDLING_CLASS_KEY=FatalErrorHelper
 - A124_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - A124_EXTENDED_ERROR_CODE=CHANGE_AMOUNT_LIMIT
- A125 Use Another Tender
 - A125_ADDITIONAL_TEXT=The use of debit tender has been temporarily suspended until problems with this function can be corrected.
 - A125_HANDLING_CLASS_KEY=FatalErrorHelper
 - A125_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A125_EXTENDED_ERROR_CODE=EXTERNAL_TENDER_AUTHORIZATION_SUSPENDED
- A126 OPID xxxxxx on Terminal nnn
 - A126_ADDITIONAL_TEXT=Operator xxxxxx is already signed on to terminal nnn.
 - A126_HANDLING_CLASS_KEY=FatalErrorHelper
 - A126_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A126_EXTENDED_ERROR_CODE=OPERATOR_ALREADY_ACTIVE
- A127 Transaction not Allowed When Online
 - A127_ADDITIONAL_TEXT=Terminal program load is not allowed online.
 - A127_HANDLING_CLASS_KEY=FatalErrorHelper
 - A127_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A127_EXTENDED_ERROR_CODE=NOT_ALLOWED_OFFLINE
- A128 (No Text Message)
 - A128_ADDITIONAL_TEXT=Terminal program load cannot complete.
 - A128_HANDLING_CLASS_KEY=FatalErrorHelper
 - A128_ERROR_CODE=POS_APP_FAILURE
 - A128_EXTENDED_ERROR_CODE=NONE
- A129 Controller is Monitoring
 - A129_ADDITIONAL_TEXT=The controller is currently monitoring a terminal.
 - A129_HANDLING_CLASS_KEY=FatalErrorHelper
 - A129_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A129_EXTENDED_ERROR_CODE=NONE
- A130 Term Monitor Active on nnn
 - A130_ADDITIONAL_TEXT=Terminal nnn is currently monitoring a terminal. Only one terminal can monitor at a time.
 - A130_HANDLING_CLASS_KEY=FatalErrorHelper
 - A130_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A130_EXTENDED_ERROR_CODE=NONE

- A131 Layaway Account has been Cancelled
 - A131_ADDITIONAL_TEXT=The layaway account number has been canceled.
 - A131_HANDLING_CLASS_KEY=FatalErrorHandler
 - A131_ERROR_CODE=INVALID_ARGUMENT
 - A131_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- A132 Bal Due = 0 Cancel not Allowed
 - A132_ADDITIONAL_TEXT=The layaway account has been paid in full and cannot be canceled.
 - A132_HANDLING_CLASS_KEY=FatalErrorHandler
 - A132_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A132_EXTENDED_ERROR_CODE=ACCOUNT_PAID_IN_FULL
- A133 Amount Entered is not Correct
 - A133_ADDITIONAL_TEXT=The amount entered for a line item cancel does not match the amount entered in the initial layaway for the item.
 - A133_HANDLING_CLASS_KEY=FatalErrorHandler
 - A133_ERROR_CODE=INVALID_ARGUMENT
 - A133_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- A134 Account has not been Updated
 - A134_ADDITIONAL_TEXT=The initial layaway transaction has not been processed by the background program. Payments or cancels cannot be performed for this account until it has been added to the file.
 - A134_HANDLING_CLASS_KEY=FatalErrorHandler
 - A134_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A134_EXTENDED_ERROR_CODE=ACCOUNT_NOT_YET_AVAILABLE
- A135 Layaway File Space Exceeded
 - A135_ADDITIONAL_TEXT=The layaway file does not have enough space left in it for the number of items being put on layaway.
 - A135_HANDLING_CLASS_KEY=FatalErrorHandler
 - A135_ERROR_CODE=POS_APP_FAILURE
 - A135_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A136 Monitoring Cannot Continue
 - A136_ADDITIONAL_TEXT=An unrecoverable error occurred during the terminal monitor procedure.
 - A136_HANDLING_CLASS_KEY=FatalErrorHandler
 - A136_ERROR_CODE=POS_APP_FAILURE
 - A136_EXTENDED_ERROR_CODE=NONE
- A137 Terminal nnn is not Responding
 - A137_ADDITIONAL_TEXT=Communication with the monitored terminal has been lost.
 - A137_HANDLING_CLASS_KEY=FatalErrorHandler
 - A137_ERROR_CODE=POS_APP_FAILURE
 - A137_EXTENDED_ERROR_CODE=NONE
- A138 Function is not Supported
 - A138_ADDITIONAL_TEXT=The function has not been selected as an option for the terminal application.
 - A138_HANDLING_CLASS_KEY=FatalErrorHandler
 - A138_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A138_EXTENDED_ERROR_CODE=NOT_AUTHORIZED
- A140 Tax Control File Mismatch
 - A140_ADDITIONAL_TEXT=The terminal has just loaded terminal options and has determined that the tax code terminal option information is in conflict with the tax control file information loaded in the terminal. This problem occurs when a new tax control file is placed on the store controller and either the appropriate tax code personalization has not been performed or the terminal has been reloaded but not both.

- A140_HANDLING_CLASS_KEY=FatalErrorHelper
 - A140_ERROR_CODE=CONFIG_ERROR
 - A140_EXTENDED_ERROR_CODE=TAX_OPTION_MISMATCH
- A162 Journal Print Required
 - A162_ADDITIONAL_TEXT=This message applies to model 3 or later printers. The journal print buffer is full. The document will be ejected and the journal buffer will print.
 - A162_HANDLING_CLASS_KEY=FatalErrorHelper
 - A162_ERROR_CODE=PRINTER_ERROR
 - A162_EXTENDED_ERROR_CODE=BUFFER_FULL
- A163 Printer Cover Open
 - A163_ADDITIONAL_TEXT=The printer cover is open.
 - A163_HANDLING_CLASS_KEY=PrinterPaperOutErrorHelper
 - A163_ERROR_CODE=PRINTER_ERROR
 - A163_EXTENDED_ERROR_CODE=COVER_OPEN
- A164 Check Paper Journal Station
 - A164_ADDITIONAL_TEXT=The journal paper station has a problem.
 - A164_HANDLING_CLASS_KEY=FatalErrorHelper
 - A164_ERROR_CODE=PRINTER_ERROR
 - A164_EXTENDED_ERROR_CODE=JOURNAL_ERROR
- A165 No Exchange Rate Found
 - A165_ADDITIONAL_TEXT=Operator has keyed a tender type that the store options specify as a foreign tender, but there is no exchange rate in the exchange file.
 - A165_HANDLING_CLASS_KEY=FatalErrorHelper
 - A165_ERROR_CODE=CONFIG_ERROR
 - A165_EXTENDED_ERROR_CODE=MISSING_EXCHANGE_RATE
- A166 Tender Type Incorrect
 - A166_ADDITIONAL_TEXT=Operator has converted the balance due to a foreign tender but then tried to tender a different tender type.
 - A166_HANDLING_CLASS_KEY=FatalErrorHelper
 - A166_ERROR_CODE=INVALID_ARGUMENT
 - A166_EXTENDED_ERROR_CODE=INVALID_TENDER_TYPE
- A167 Unable to Load Item Validation File
 - A167_ADDITIONAL_TEXT=The system is unable to load the Specific Item Validation File for this terminal.
 - A167_HANDLING_CLASS_KEY=InformationalErrorHelper
 - A167_ERROR_CODE=POS_APP_FAILURE
 - A167_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A168 Default Options will be Used
 - A168_ADDITIONAL_TEXT=This is a part of the previous message. This message indicates that the default options will be used if the terminal is allowed to complete the load. This depends on store policy.
 - A168_HANDLING_CLASS_KEY=InformationalErrorHelper
 - A168_ERROR_CODE=NONE
 - A168_EXTENDED_ERROR_CODE=NONE
- A169 Unknown Account Format
 - A169_ADDITIONAL_TEXT=The magnetic ink character recognition/reader (MICR) reading the module does not recognize the MICR format on the check. This indicates a problem with the MICR parsing routine.
 - A169_HANDLING_CLASS_KEY=FatalErrorHelper
 - A169_ERROR_CODE=CONFIG_ERROR
 - A169_EXTENDED_ERROR_CODE=UNKNOWN_MICR_FORMAT
- A170 Error Reading Check Account
 - A170_ADDITIONAL_TEXT=The MICR reading module failed to capture the account number.

- A170_HANDLING_CLASS_KEY=FatalErrorHelper
 - A170_ERROR_CODE=MICR_ERROR
 - A170_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_ERROR
- A175 Can't Access Policy From Help
 - A175_ADDITIONAL_TEXT=Policy screens cannot be accessed from help.
 - A175_HANDLING_CLASS_KEY=FatalErrorHelper
 - A175_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A175_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A176 Name is Required
 - A176_ADDITIONAL_TEXT=Name was not entered as required by store options.
 - A176_HANDLING_CLASS_KEY=FatalErrorHelper
 - A176_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A176_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A177 Address is Required
 - A177_ADDITIONAL_TEXT=Address was not entered as required by store options.
 - A177_HANDLING_CLASS_KEY=FatalErrorHelper
 - A177_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A177_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A178 City is Required
 - A178_ADDITIONAL_TEXT=City was not entered as required by store options.
 - A178_HANDLING_CLASS_KEY=FatalErrorHelper
 - A178_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A178_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A179 State is Required
 - A179_ADDITIONAL_TEXT=State was not entered as required by store options.
 - A179_HANDLING_CLASS_KEY=FatalErrorHelper
 - A179_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A179_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A180 Postal Code is Required
 - A180_ADDITIONAL_TEXT=Zip code was not entered as required by store options.
 - A180_HANDLING_CLASS_KEY=FatalErrorHelper
 - A180_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A180_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A181 Phone is Required
 - A181_ADDITIONAL_TEXT=Phone number was not entered as required by store options.
 - A181_HANDLING_CLASS_KEY=FatalErrorHelper
 - A181_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A181_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A182 Additional Info is Required
 - A182_ADDITIONAL_TEXT=Additional information was not entered as required by store options.
 - A182_HANDLING_CLASS_KEY=FatalErrorHelper
 - A182_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A182_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A183 Please try Message Later
 - A183_ADDITIONAL_TEXT=Message facility is busy.
 - A183_HANDLING_CLASS_KEY=FatalErrorHelper
 - A183_ERROR_CODE=SYSTEM_BUSY
 - A183_EXTENDED_ERROR_CODE=NONE
- A184 Transaction Type Disallows Sends

- A184_ADDITIONAL_TEXT=Sends are not allowed with this transaction type.
 - A184_HANDLING_CLASS_KEY=FatalErrorHandler
 - A184_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A184_EXTENDED_ERROR_CODE=NONE
- A185 Must Page to Item to be Voided
 - A185_ADDITIONAL_TEXT=The item being voided must be on the current screen.
 - A185_HANDLING_CLASS_KEY=FatalErrorHandler
 - A185_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A185_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A186 Must Void Item's Last Occurrence
 - A186_ADDITIONAL_TEXT=Items must be voided in reverse of order entered.
 - A186_HANDLING_CLASS_KEY=FatalErrorHandler
 - A186_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A186_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A187 This Item is Already Voided
 - A187_ADDITIONAL_TEXT=A void was requested for an item that was already voided.
 - A187_HANDLING_CLASS_KEY=FatalErrorHandler
 - A187_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A187_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A188 Name is Invalid
 - A188_ADDITIONAL_TEXT=This message is the result of user exit processing. Editing by user exit EALFSUIO has determined that the name is invalid.
 - A188_HANDLING_CLASS_KEY=FatalErrorHandler
 - A188_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A188_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A189 Address is Invalid
 - A189_ADDITIONAL_TEXT=This message is the result of user exit processing. Editing by user exit EALFSUIO has determined that the address is invalid.
 - A189_HANDLING_CLASS_KEY=FatalErrorHandler
 - A189_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A189_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A190 City is Invalid
 - A190_ADDITIONAL_TEXT=This message is the result of user exit processing. Editing by user exit EALFSUIO has determined that the city is invalid.
 - A190_HANDLING_CLASS_KEY=FatalErrorHandler
 - A190_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A190_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A191 State is Invalid
 - A191_ADDITIONAL_TEXT=This message is the result of user exit processing. Editing by user exit EALFSUIO has determined that the state is invalid.
 - A191_HANDLING_CLASS_KEY=FatalErrorHandler
 - A191_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A191_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A192 Postal Code is Invalid
 - A192_ADDITIONAL_TEXT=This message is the result of user exit processing. Editing by user exit EALFSUIO has determined that the postal code is invalid.
 - A192_HANDLING_CLASS_KEY=FatalErrorHandler
 - A192_ERROR_CODE=PROCEDURE_NOT_ALLOWED

- A192_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A193 Phone is Invalid
 - A193_ADDITIONAL_TEXT=This message is the result of user exit processing. Editing by user exit EALFSUIO has determined that the phone is invalid.
 - A193_HANDLING_CLASS_KEY=FatalErrorHelper
 - A193_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A193_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A194 Only 7 Recipients can be Specified
 - A194_ADDITIONAL_TEXT=The maximum number of send recipients is seven.
 - A194_HANDLING_CLASS_KEY=FatalErrorHelper
 - A194_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A194_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A195 Enter Manual Tax Amount
 - A195_ADDITIONAL_TEXT=A manual tax code was selected but no amount has been entered.
 - A195_HANDLING_CLASS_KEY=FatalErrorHelper
 - A195_ERROR_CODE=INVALID_ARGUMENT
 - A195_EXTENDED_ERROR_CODE=AMOUNT_REQUIRED
- A196 Page Down is not Valid
 - A196_ADDITIONAL_TEXT=You cannot page down from this screen.
 - A196_HANDLING_CLASS_KEY=FatalErrorHelper
 - A196_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A196_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A197 Page Up is not Valid
 - A197_ADDITIONAL_TEXT=You cannot page up from this screen.
 - A197_HANDLING_CLASS_KEY=FatalErrorHelper
 - A197_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A197_EXTENDED_ERROR_CODE=FULLSCREEN_NOT_SUPPORTED
- A198 Void not Allowed After Send Item
 - A198_ADDITIONAL_TEXT=Void of a send item is not allowed.
 - A198_HANDLING_CLASS_KEY=FatalErrorHelper
 - A198_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A198_EXTENDED_ERROR_CODE=NONE
- A200 Translation Program Problem - Check System Log
 - A200_ADDITIONAL_TEXT=An error has occurred in a translation function.
 - A200_HANDLING_CLASS_KEY=FatalErrorHelper
 - A200_ERROR_CODE=NONE
 - A200_EXTENDED_ERROR_CODE=NONE
- A201 Display Program Problem - Check System Log
 - A201_ADDITIONAL_TEXT=An error occurred during the execution of a Display Manager function.
 - A201_HANDLING_CLASS_KEY=FatalErrorHelper
 - A201_ERROR_CODE=NONE
 - A201_EXTENDED_ERROR_CODE=NONE
- A202 Record not Added, File Space Exceeded- Check System Log
 - A202_ADDITIONAL_TEXT=An error occurred trying to add a record to the file and space was not available.
 - A202_HANDLING_CLASS_KEY=FatalErrorHelper
 - A202_ERROR_CODE=NONE
 - A202_EXTENDED_ERROR_CODE=NONE
- A203 Screen Defaults File was not Found - Check System Log
 - A203_ADDITIONAL_TEXT=The Application Screen Defaults file is not on the file directory.
 - A203_HANDLING_CLASS_KEY=FatalErrorHelper

- A203_ERROR_CODE=NONE
 - A203_EXTENDED_ERROR_CODE=NONE
- A204 Unexpected Error Occurred Processing the Transaction Log - Check System Log
 - A204_ADDITIONAL_TEXT=An error occurred while processing the transaction log (EALTRANS).
 - A204_HANDLING_CLASS_KEY=FatalErrorHandler
 - A204_ERROR_CODE=NONE
 - A204_EXTENDED_ERROR_CODE=NONE
- A205 Information for Report is not Available
 - A205_ADDITIONAL_TEXT=An automated procedure requested a report for which there was no data. (The file had been reset before the report request.)
 - A205_HANDLING_CLASS_KEY=FatalErrorHandler
 - A205_ERROR_CODE=NONE
 - A205_EXTENDED_ERROR_CODE=NONE
- A206 Error from Application Services
 - A206_ADDITIONAL_TEXT=A call to application services generated an unexpected return code. Necessary information must be obtained from an application services call in the retail application main menu routine. Controller functions selected from the application main menu will not execute until this call has completed successfully.
 - A206_HANDLING_CLASS_KEY=FatalErrorHandler
 - A206_ERROR_CODE=NONE
 - A206_EXTENDED_ERROR_CODE=NONE
- A207 Host Credit Pipe Error - Check System Log
 - A207_ADDITIONAL_TEXT=An error occurred while writing to the terminal response pipe.
 - A207_HANDLING_CLASS_KEY=FatalErrorHandler
 - A207_ERROR_CODE=NONE
 - A207_EXTENDED_ERROR_CODE=NONE
- A208 DDM Ended - Error in "Start Immediate" Task
 - A208_ADDITIONAL_TEXT=The delayed data maintenance start immediate task has terminated due to an error.
 - A208_HANDLING_CLASS_KEY=FatalErrorHandler
 - A208_ERROR_CODE=NONE
 - A208_EXTENDED_ERROR_CODE=NONE
- A210 DDM Ended - Error in Background Task
 - A210_ADDITIONAL_TEXT=DDM Ended - Error in Background Task.
 - A210_HANDLING_CLASS_KEY=FatalErrorHandler
 - A210_ERROR_CODE=NONE
 - A210_EXTENDED_ERROR_CODE=NONE
- A211 DDM Ended - Error in Timer Task
 - A211_ADDITIONAL_TEXT=The delayed data maintenance timer task has terminated due to an error.
 - A211_HANDLING_CLASS_KEY=FatalErrorHandler
 - A211_ERROR_CODE=NONE
 - A211_EXTENDED_ERROR_CODE=NONE
- A212 Error Accessing File
 - A212_ADDITIONAL_TEXT=An unexpected error occurred during a file access.
 - A212_HANDLING_CLASS_KEY=FatalErrorHandler
 - A212_ERROR_CODE=POS_APP_FAILURE
 - A212_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- A221 EFT Device is Loading
 - A221_ADDITIONAL_TEXT=The EFT device is loading and therefore EFT tender is not allowed as payment during this transaction.

- A221_HANDLING_CLASS_KEY=FatalErrorHandler
 - A221_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A221_EXTENDED_ERROR_CODE=PINPAD_NOT_AVAILABLE
- A222 Check EFT Device
 - A222_ADDITIONAL_TEXT=There is a problem with the EFT device.
 - A222_HANDLING_CLASS_KEY=FatalErrorHandler
 - A222_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - A222_EXTENDED_ERROR_CODE=PINPAD_NOT_AVAILABLE
- A223 Account Number Read from Card
 - A223_ADDITIONAL_TEXT=An account number was keyed at the terminal and an account number was also read at the EFT device. The account number read at the EFT device will be used.
 - A223_HANDLING_CLASS_KEY=InformationalErrorHandler
 - A223_ERROR_CODE=NONE
 - A223_EXTENDED_ERROR_CODE=NONE
- A224 Invalid Expiry Date
 - A224_ADDITIONAL_TEXT=The expiration date keyed at the terminal is invalid. The date must be formatted MMY.
 - A224_HANDLING_CLASS_KEY=FatalErrorHandler
 - A224_ERROR_CODE=INVALID_ARGUMENT
 - A224_EXTENDED_ERROR_CODE=INVALID_EXPIRY_DATE
- A399 Age Restricted Item
 - A399_ADDITIONAL_TEXT=The item is flagged as age restricted in the item record file. Follow the instructions on the screen.
 - A399_HANDLING_CLASS_KEY=AgeRestrictedHelper
 - A399_ERROR_CODE=AGE_RESTRICTED_ITEM
 - A399_EXTENDED_ERROR_CODE=NONE
- BASE_DESC35 Insert Document
 - BASE_DESC35_ADDITIONAL_TEXT=Insert Document into the DI station and press the Clear key.
 - BASE_DESC35_HANDLING_CLASS_KEY=PrinterErrorHandler
 - BASE_DESC35_ERROR_CODE=NONE
 - BASE_DESC35_EXTENDED_ERROR_CODE=NONE
- BASE_DESC36 Insert Document for Franking
 - BASE_DESC36_ADDITIONAL_TEXT=Insert Document into the DI station for franking and press the Clear key.
 - BASE_DESC36_HANDLING_CLASS_KEY=InsertDocumentHelper
 - BASE_DESC36_ERROR_CODE=NONE
 - BASE_DESC36_EXTENDED_ERROR_CODE=NONE
- BASE_DESC37 Remove Document
 - BASE_DESC37_ADDITIONAL_TEXT=Remove the Document from the DI station.
 - BASE_DESC37_HANDLING_CLASS_KEY=RemoveDocumentHelper
 - BASE_DESC37_ERROR_CODE=NONE
 - BASE_DESC37_EXTENDED_ERROR_CODE=NONE
- BASE_DESC45 Terminal Disabled
 - BASE_DESC45_ADDITIONAL_TEXT=This terminal has been disabled. Please enable the terminal and try again.
 - BASE_DESC45_HANDLING_CLASS_KEY=TerminalDisabledErrorHandler
 - BASE_DESC45_ERROR_CODE=TERMINAL_DISABLED
 - BASE_DESC45_EXTENDED_ERROR_CODE=NONE
- BASE_DESC255 Insert Cheque to be Printed
 - BASE_DESC255_ADDITIONAL_TEXT=Insert Check into the DI station and press the Clear key.
 - BASE_DESC255_HANDLING_CLASS_KEY=PrinterErrorHandler
 - BASE_DESC255_ERROR_CODE=NONE

- BASE_DESC255_EXTENDED_ERROR_CODE=NONE
- BASE_DESC449 Blank Line on the Display
 - BASE_DESC449_ADDITIONAL_TEXT=There is a blank line on the display while in the error helper.
 - BASE_DESC449_HANDLING_CLASS_KEY=BaseDescriptor449ErrorHandler
 - BASE_DESC449_ERROR_CODE=ERROR_HANDLER_OTHER_ERROR
 - BASE_DESC449_EXTENDED_ERROR_CODE=NONE
- STATE_6 Enter Authorization Number
 - STATE_6_ADDITIONAL_TEXT=Enter the authorization number followed by the Enter key.
 - STATE_6_HANDLING_CLASS_KEY=EnterAuthorizationErrorHandler
 - STATE_6_ERROR_CODE=ENABLEMENT_ERROR
 - STATE_6_EXTENDED_ERROR_CODE=AUTHORIZATION_NUMBER_MISMATCH
- STATE_10 Enter Quantity or Price
 - STATE_10_ADDITIONAL_TEXT=Please enter the Quantity or Price for this item.
 - STATE_10_HANDLING_CLASS_KEY=QuantityOrPriceErrorHandler
 - STATE_10_ERROR_CODE=INVALID_ARGUMENT
 - STATE_10_EXTENDED_ERROR_CODE=ITEM_PRICE_OR_QUANTITY_REQUIRED
- STATE_20 Original Salesperson is Required
 - STATE_20_ADDITIONAL_TEXT=The terminal options indicates that the original salesperson is required to return an item.
 - STATE_20_HANDLING_CLASS_KEY=OriginalSalespersonRequiredErrorHandler
 - STATE_20_ERROR_CODE=INVALID_ARGUMENT
 - STATE_20_EXTENDED_ERROR_CODE=ID_REQUIRED
- STATE_49 Quantity is Required
 - STATE_49_ADDITIONAL_TEXT=The Item Record File indicates that a quantity is required for this item.
 - STATE_49_HANDLING_CLASS_KEY=QuantityErrorHandler
 - STATE_49_ERROR_CODE=INVALID_ARGUMENT
 - STATE_49_EXTENDED_ERROR_CODE=ITEM_QUANTITY_REQUIRED
- STATE_50 Price is Required
 - STATE_50_ADDITIONAL_TEXT=The Item Record File indicates that a price is required for this item.
 - STATE_50_HANDLING_CLASS_KEY=PriceRequiredErrorHandler
 - STATE_50_ERROR_CODE=INVALID_ARGUMENT
 - STATE_50_EXTENDED_ERROR_CODE=ITEM_PRICE_REQUIRED
- STATE_51 Enter Account Number
 - STATE_51_ADDITIONAL_TEXT=Enter the account number followed by the Enter key.
 - STATE_51_HANDLING_CLASS_KEY=AccountNumberErrorHandler
 - STATE_51_ERROR_CODE=INVALID_ARGUMENT
 - STATE_51_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_REQUIRED
- STATE_94 Enter to Print or Clear to Bypass
 - STATE_94_ADDITIONAL_TEXT=Press the Enter key to print or the Clear key to bypass printing.
 - STATE_94_HANDLING_CLASS_KEY=PrintBypassErrorHandler
 - STATE_94_ERROR_CODE=NONE
 - STATE_94_EXTENDED_ERROR_CODE=NONE
- STATE_97 Enter to Accept or Clear to Void
 - STATE_97_ADDITIONAL_TEXT=Press the Enter key to Accept or the Clear key to Void tender.

- STATE_97_HANDLING_CLASS_KEY=AcceptOrVoidTender
 - STATE_97_ERROR_CODE=TENDER_NOT_ACCEPTED
 - STATE_97_EXTENDED_ERROR_CODE=CANCELLED_BY_OPERATOR
- 4610_A004 Printer Out of Paper or Cover Open
 - 4610_A004_ADDITIONAL_TEXT=The printer customer receipt station is out of paper or the station cover has been opened.
 - 4610_A004_HANDLING_CLASS_KEY=PrinterPaperOutErrorHandler
 - 4610_A004_ERROR_CODE=PRINTER_ERROR
 - 4610_A004_EXTENDED_ERROR_CODE=NONE
- 4610_A005 Invalid Data
 - 4610_A005_ADDITIONAL_TEXT=The printer has received a control that it does not recognize.
 - 4610_A005_HANDLING_CLASS_KEY=PrinterErrorHandler
 - 4610_A005_ERROR_CODE=POS_APP_FAILURE
 - 4610_A005_EXTENDED_ERROR_CODE=UNRECOGNIZED_PRINT_CHARACTERS
- 95_APPROVE_OR_VOID__ Prompt to Approve or Void a Credit Transaction
 - 95_APPROVE_OR_VOID__ADDITIONAL_TEXT=This tender needs to be approved. Press the Clear key to continue.
 - 95_APPROVE_OR_VOID__HANDLING_CLASS_KEY=InformationalErrorHandler
 - 95_APPROVE_OR_VOID__ERROR_CODE=NONE
 - 95_APPROVE_OR_VOID__EXTENDED_ERROR_CODE=NONE

nextFile=usererror – This property is used to tell AEF the name of the next property file in the bundle chain.

Apperror.properties (SA)

The apperror.properties file includes the following properties.

- Default error helper used when the key is not found in the error bundle or no helper class is specified.
 - APP_DEFAULT_ADDITIONAL_TEXT=The Default error helper was used because this error or its helper class is not defined in the error property files.
 - APP_DEFAULT_HANDLING_CLASS_KEY=AppDefaultErrorHandler
 - APP_DEFAULT_ERROR_CODE=ERROR_HANDLER_OTHER_ERROR
 - APP_DEFAULT_EXTENDED_ERROR_CODE=NONE
- B000 Abnormal Application End
 - B000_ADDITIONAL_TEXT=An unrecoverable terminal error has been detected.
 - B000_HANDLING_CLASS_KEY=FatalErrorHandler
 - B000_ERROR_CODE=POS_APP_FAILURE
 - B000_EXTENDED_ERROR_CODE=NONE
- B001 No Coupon on Item Description
 - B001_ADDITIONAL_TEXT=A coupon key was pressed for an item code which cannot be used as a coupon.
 - B001_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B001_ERROR_CODE=INVALID_ARGUMENT
 - B001_EXTENDED_ERROR_CODE=INVALID_ITEM_CODE
- B002 System Busy, Wait and Retry
 - B002_ADDITIONAL_TEXT=The system has not finished processing previous entries. The entry was not accepted by the system.
 - B002_HANDLING_CLASS_KEY=FatalErrorHandler
 - B002_ERROR_CODE=SYSTEM_BUSY

- B002_EXTENDED_ERROR_CODE=NONE
- B003 Check Key Sequence
 - B003_ADDITIONAL_TEXT=The key sequence which was entered is invalid or cannot be used at this time.
 - B003_HANDLING_CLASS_KEY=FatalErrorHandler
 - B003_ERROR_CODE=CONFIG_ERROR
 - B003_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B003 Check Keyed Label Data
 - B003_KEYED_LABEL_ADDITIONAL_TEXT=The keyed label data is invalid. Check the check digit.
 - B003_KEYED_LABEL_HANDLING_CLASS_KEY=FatalErrorHandler
 - B003_KEYED_LABEL_ERROR_CODE=INVALID_ARGUMENT
 - B003_KEYED_LABEL_EXTENDED_ERROR_CODE=AEFConst.NONE
- B004 Managers Key is Required
 - B004_ADDITIONAL_TEXT=The manager's key is required to perform a manager's override.
 - B004_HANDLING_CLASS_KEY=SAErrorHandlerB004
 - B004_ERROR_CODE=KEYLOCK_ERROR
 - B004_EXTENDED_ERROR_CODE=MANAGER_KEY_REQUIRED
- B005 Remove Manager's Key
 - B005_ADDITIONAL_TEXT=The manager's key must be removed after being used with a manager's override sequence.
 - B005_HANDLING_CLASS_KEY=SAErrorHandlerB005
 - B005_ERROR_CODE=KEYLOCK_ERROR
 - B005_EXTENDED_ERROR_CODE=MANAGER_KEY_NOT_REMOVED
- B006 Keyed Data Out of Range
 - B006_ADDITIONAL_TEXT=A numeric entry in the key sequence is too large or too small.
 - B006_HANDLING_CLASS_KEY=FatalErrorHandler
 - B006_ERROR_CODE=INVALID_ARGUMENT
 - B006_EXTENDED_ERROR_CODE=DATA_OUT_OF_RANGE
- B007 Check the Printer
 - B007_ADDITIONAL_TEXT=A Printer Error has Occurred.
 - B007_HANDLING_CLASS_KEY=PrinterErrorHandler
 - B007_ERROR_CODE=PRINTER_ERROR
 - B007_EXTENDED_ERROR_CODE=NONE
- B008 Transaction Limit
 - B008_ADDITIONAL_TEXT=A transaction limit has been exceeded.
 - B008_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B008_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B008_EXTENDED_ERROR_CODE=TRANSACTION_LIMIT
- B009 Item Limit Check
 - B009_ADDITIONAL_TEXT=An item limit has been exceeded.
 - B009_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B009_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B009_EXTENDED_ERROR_CODE=ITEM_LIMIT
- B010 Use Lookup Keys
 - B010_ADDITIONAL_TEXT=The Item Record File cannot be successfully accessed.
 - B010_HANDLING_CLASS_KEY=FatalErrorHandler
 - B010_ERROR_CODE=POS_APP_FAILURE
 - B010_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B011 Check Coin Dispenser
 - B011_ADDITIONAL_TEXT=An error has been detected on the coin dispenser.
 - B011_HANDLING_CLASS_KEY=InformationalErrorHandler

- B011_ERROR_CODE=COIN_DISPENSER_ERROR
 - B011_EXTENDED_ERROR_CODE=NONE
- B012 Check Journal Paper
 - B012_ADDITIONAL_TEXT=A paper-low condition has been detected on the Transaction Summary Journal.
 - B012_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B012_ERROR_CODE=PRINTER_ERROR
 - B012_EXTENDED_ERROR_CODE=PAPER_LOW
- B013 Close the Cash Drawer
 - B013_ADDITIONAL_TEXT=The cash drawer must be closed.
 - B013_HANDLING_CLASS_KEY=SAErrorHelperB013
 - B013_ERROR_CODE=CASH_DRAWER_ERROR
 - B013_EXTENDED_ERROR_CODE=CLOSE_DRAWER
- B014 No Transaction Recovery
 - B014_ADDITIONAL_TEXT=A transaction in progress has not been recovered following a terminal restart or terminal transfer.
 - B014_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B014_ERROR_CODE=NONE
 - B014_EXTENDED_ERROR_CODE=NONE
- B015 Check the Keyboard
 - B015_ADDITIONAL_TEXT=The application program failed in an attempt to reach the beeper in the keyboard.
 - B015_HANDLING_CLASS_KEY=FatalErrorHelper
 - B015_ERROR_CODE=KEYBOARD_ERROR
 - B015_EXTENDED_ERROR_CODE=NONE
- B016 Write Error - Transaction Lost
 - B016_ADDITIONAL_TEXT=A hardware error caused a write failure in the Transaction Summary Log and the completed transaction is lost.
 - B016_HANDLING_CLASS_KEY=FatalErrorHelper
 - B016_ERROR_CODE=POS_APP_FAILURE
 - B016_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B017 Extended Price Limit Check
 - B017_ADDITIONAL_TEXT=The product of quantity and price cannot exceed eight digits for any item entry.
 - B017_HANDLING_CLASS_KEY=FatalErrorHelper
 - B017_ERROR_CODE=INVALID_ARGUMENT
 - B017_EXTENDED_ERROR_CODE=EXTENDED_PRICE_TOO_LARGE
- B018 No Weight on Item Description
 - B018_ADDITIONAL_TEXT=The Weight or Tare key was pressed for an item which is not sold by weight.
 - B018_HANDLING_CLASS_KEY=FatalErrorHelper
 - B018_ERROR_CODE=INVALID_ARGUMENT
 - B018_EXTENDED_ERROR_CODE=ITEM_WEIGHT_PROHIBITED
- B019 Weight Needed on Item Description
 - B019_ADDITIONAL_TEXT=A weight or price must be entered for an item which is sold by weight.
 - B019_HANDLING_CLASS_KEY=ItemPromptErrorHelper
 - B019_ERROR_CODE=INVALID_ARGUMENT
 - B019_EXTENDED_ERROR_CODE=ITEM_WEIGHT_REQUIRED
- B020 Take Total
 - B020_ADDITIONAL_TEXT=A transaction total is required.
 - B020_HANDLING_CLASS_KEY=FatalErrorHelper
 - B020_ERROR_CODE=CONFIG_ERROR
 - B020_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B021 Check Tare Code

- B021_ADDITIONAL_TEXT=The tare code entered with the Tare key is not defined.
 - B021_HANDLING_CLASS_KEY=FatalErrorHandler
 - B021_ERROR_CODE=INVALID_ARGUMENT
 - B021_EXTENDED_ERROR_CODE=INVALID_TARE
- B022 No Deposit on Item Description
 - B022_ADDITIONAL_TEXT=The Deposit key has been pressed for an item code which cannot be used as a deposit.
 - B022_HANDLING_CLASS_KEY=FatalErrorHandler
 - B022_ERROR_CODE=INVALID_ARGUMENT
 - B022_EXTENDED_ERROR_CODE=INVALID_ITEM_CODE
- B023 No Quantity on Item Description
 - B023_ADDITIONAL_TEXT=The Quantity key was pressed for an item which is sold by weight or does not allow for quantity entry.
 - B023_HANDLING_CLASS_KEY=FatalErrorHandler
 - B023_ERROR_CODE=INVALID_ARGUMENT
 - B023_EXTENDED_ERROR_CODE=ITEM_QUANTITY_PROHIBITED
- B024 PRICE NEEDED item description
 - B024_ADDITIONAL_TEXT=A price must be entered for this item.
 - B024_HANDLING_CLASS_KEY=ItemPromptErrorHandler
 - B024_ERROR_CODE=INVALID_ARGUMENT
 - B024_EXTENDED_ERROR_CODE=ITEM_PRICE_REQUIRED
- B025 Sell Item Without Repeat Key
 - B025_ADDITIONAL_TEXT=The limit for consecutive repeat entries using a repeat key has been exceeded.
 - B025_HANDLING_CLASS_KEY=FatalErrorHandler
 - B025_ERROR_CODE=CONFIG_ERROR
 - B025_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B026 Item Not Found Item Code
 - B026_ADDITIONAL_TEXT=The displayed item code is not on the Item Record file.
 - B026_HANDLING_CLASS_KEY=FatalErrorHandler
 - B026_ERROR_CODE=ITEM_NOT_FOUND
 - B026_EXTENDED_ERROR_CODE=NONE
- B027 Not for Sale Item Description
 - B027_ADDITIONAL_TEXT=The item is not for sale or is not allowed to be sold during selected time periods.
 - B027_HANDLING_CLASS_KEY=FatalErrorHandler
 - B027_ERROR_CODE=ITEM_NOT_FOR_SALE
 - B027_EXTENDED_ERROR_CODE=NONE
- B028 Take Food Stamp Total
 - B028_ADDITIONAL_TEXT=A food stamp total is required before a food stamp tender.
 - B028_HANDLING_CLASS_KEY=FatalErrorHandler
 - B028_ERROR_CODE=CONFIG_ERROR
 - B028_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B029 Quantity Needed Item Description
 - B029_ADDITIONAL_TEXT=A quantity must be entered for this item.
 - B029_HANDLING_CLASS_KEY=ItemPromptErrorHandler
 - B029_ERROR_CODE=INVALID_ARGUMENT
 - B029_EXTENDED_ERROR_CODE=ITEM_QUANTITY_REQUIRED
- B030 Transaction Total is Too Large
 - B030_ADDITIONAL_TEXT=The balance due has exceeded the 9 digit upper limit of the system.
 - B030_HANDLING_CLASS_KEY=FatalErrorHandler
 - B030_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED

- B030_EXTENDED_ERROR_CODE=TRANSACTION_TOTAL_TOO_LARGE
- B031 No FS Key on Item Description
 - B031_ADDITIONAL_TEXT=The FS/No FS or Food Stamp keys are only allowed with selected department keys.
 - B031_HANDLING_CLASS_KEY=FatalErrorHandler
 - B031_ERROR_CODE=CONFIG_ERROR
 - B031_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B032 No Tax Key on Item Description
 - B032_ADDITIONAL_TEXT=The Tax/No Tax key is only allowed with selected department keys.
 - B032_HANDLING_CLASS_KEY=FatalErrorHandler
 - B032_ERROR_CODE=CONFIG_ERROR
 - B032_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B033 Scale Error Clear/Retry
 - B033_ADDITIONAL_TEXT=The scale driver detected bad data from the interface, or a read of the interface occurred
 - B033_HANDLING_CLASS_KEY=FatalErrorHandler
 - B033_ERROR_CODE=SCALE_ERROR
 - B033_EXTENDED_ERROR_CODE=NONE
- B034 Check Tender Variety
 - B034_ADDITIONAL_TEXT=The tender variety entered with the asterisk (*) key is not defined.
 - B034_HANDLING_CLASS_KEY=FatalErrorHandler
 - B034_ERROR_CODE=CONFIG_ERROR
 - B034_EXTENDED_ERROR_CODE=UNDEFINED_TENDER_VARIETY
- B035 Check Food Stamp Denomination
 - B035_ADDITIONAL_TEXT=Food Stamps can only be tendered in dollar denominations.
 - B035_HANDLING_CLASS_KEY=FatalErrorHandler
 - B035_ERROR_CODE=INVALID_ARGUMENT
 - B035_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- B036 Check Quantity
 - B036_ADDITIONAL_TEXT=The entered quantity displayed in the guidance message may be due to a keying error.
 - B036_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B036_ERROR_CODE=INVALID_ARGUMENT
 - B036_EXTENDED_ERROR_CODE=INVALID_QUANTITY
- B037 Check Price
 - B037_ADDITIONAL_TEXT=The tendered price displayed in the guidance message is likely the result of a keying error.
 - B037_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B037_ERROR_CODE=INVALID_ARGUMENT
 - B037_EXTENDED_ERROR_CODE=INVALID_PRICE
- B038 Check Weight
 - B038_ADDITIONAL_TEXT=The entered weight displayed in the guidance message is likely the result of a keying error.
 - B038_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B038_ERROR_CODE=INVALID_ARGUMENT
 - B038_EXTENDED_ERROR_CODE=INVALID_WEIGHT
- B039 Check Item Data
 - B039_ADDITIONAL_TEXT=The item record for the entered item contains invalid data.
 - B039_HANDLING_CLASS_KEY=FatalErrorHandler
 - B039_ERROR_CODE=POS_APP_FAILURE
 - B039_EXTENDED_ERROR_CODE=INVALID_APPLICATION_DATA

- B040 File Access Failed session number
 - B040_ADDITIONAL_TEXT=A required file access has failed.
 - B040_HANDLING_CLASS_KEY=SAErrorHelperB040
 - B040_ERROR_CODE=POS_APP_FAILURE
 - B040_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B041 Check Account Data
 - B041_ADDITIONAL_TEXT=The tender verification record for the entered account number contains invalid status data.
 - B041_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B041_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B041_EXTENDED_ERROR_CODE=TENDER_NOT_AUTHORIZED
- B042 Item is on the Scale
 - B042_ADDITIONAL_TEXT=A weight cannot be keyed while an item is on the scale.
 - B042_HANDLING_CLASS_KEY=FatalErrorHelper
 - B042_ERROR_CODE=SCALE_ERROR
 - B042_EXTENDED_ERROR_CODE=ITEM_IS_ON_THE_SCALE
- B043 No Refund on Item Description
 - B043_ADDITIONAL_TEXT=The Refund key has been pressed for an item code which cannot be refunded or for a Department key which is not allowed to be used with the Refund key.
 - B043_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B043_ERROR_CODE=ITEM_NOT_RETURNABLE
 - B043_EXTENDED_ERROR_CODE=NONE
- B044 Not Available Online
 - B044_ADDITIONAL_TEXT=The TPL procedure has been selected while still online. This procedure is only available off-line.
 - B044_HANDLING_CLASS_KEY=FatalErrorHelper
 - B044_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B044_EXTENDED_ERROR_CODE=NONE
- B045 Discount Limit Check
 - B045_ADDITIONAL_TEXT=A discount limit has been exceeded.
 - B045_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B045_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B045_EXTENDED_ERROR_CODE=DISCOUNT_LIMIT
- B046 One Discount per Transaction
 - B046_ADDITIONAL_TEXT=Only one discount may be taken during any transaction.
 - B046_HANDLING_CLASS_KEY=FatalErrorHelper
 - B046_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B046_EXTENDED_ERROR_CODE=DISCOUNT_LIMIT
- B047 Check Discount Group Number Group
 - B047_ADDITIONAL_TEXT=The discount group number entered and displayed with the guidance is not defined.
 - B047_HANDLING_CLASS_KEY=FatalErrorHelper
 - B047_ERROR_CODE=INVALID_ARGUMENT
 - B047_EXTENDED_ERROR_CODE=INVALID_DISCOUNT_CODE
- B048 Account Number Needed
 - B048_ADDITIONAL_TEXT=An account number is needed with this tender for verification to be performed.
 - B048_HANDLING_CLASS_KEY=AccountNumberErrorHelper
 - B048_ERROR_CODE=INVALID_ARGUMENT
 - B048_EXTENDED_ERROR_CODE=ACCOUNT_NUMBER_REQUIRED
- B049 Insert Document to be franked
 - B049_ADDITIONAL_TEXT=A document is required in the document insert station for printing.

- B049_HANDLING_CLASS_KEY=PrinterErrorHandler
 - B049_ERROR_CODE=NONE
 - B049_EXTENDED_ERROR_CODE=NONE
- B050 Tender is Verified
 - B050_ADDITIONAL_TEXT=A tender entry submitted outside of a transaction for verification without cashing has been accepted.
 - B050_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B050_ERROR_CODE=NONE
 - B050_EXTENDED_ERROR_CODE=NONE
- B051 Not on File Account Number
 - B051_ADDITIONAL_TEXT=The account number entered and displayed in the guidance message cannot be verified because it is not in the verification file.
 - B051_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B051_ERROR_CODE=INVALID_ARGUMENT
 - B051_EXTENDED_ERROR_CODE=INVALID_ACCOUNT_NUMBER
- B052 Account Limit Exceeded Quantity
 - B052_ADDITIONAL_TEXT=Too many tenders have been made against this account number or in this transaction.
 - B052_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B052_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B052_EXTENDED_ERROR_CODE=ACCOUNT_TENDER_LIMIT
- B053 Account Limit Exceeded Amount
 - B053_ADDITIONAL_TEXT=The tender amount entered and shown in the guidance message exceeds the limit for this account number.
 - B053_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B053_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B053_EXTENDED_ERROR_CODE=ACCOUNT_TENDER_LIMIT
- B054 Account Limit Exceeded Total Amount
 - B054_ADDITIONAL_TEXT=The total amount of the tenders made against this account number exceeds the limit. The total for this Account is shown in the guidance message.
 - B054_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B054_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B054_EXTENDED_ERROR_CODE=ACCOUNT_TENDER_LIMIT
- B055 Risk 1
 - B055_ADDITIONAL_TEXT=The tender is rejected because the account is risk 1 as defined by your store procedures.
 - B055_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B055_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B055_EXTENDED_ERROR_CODE=RISK_1
- B056 Risk 2
 - B056_ADDITIONAL_TEXT=The tender is rejected because the account is risk 2 as defined by your store procedures.
 - B056_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B056_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B056_EXTENDED_ERROR_CODE=RISK_2
- B057 Risk 3
 - B057_ADDITIONAL_TEXT=The tender is rejected because the account is risk 3 as defined by your store procedures.
 - B057_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B057_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B057_EXTENDED_ERROR_CODE=RISK_3
- B058 Risk 4
 - B058_ADDITIONAL_TEXT=The tender is rejected because the account is risk 4 as defined by your store procedures.

- B058_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B058_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B058_EXTENDED_ERROR_CODE=RISK_4
- B059 Check Override Number
 - B059_ADDITIONAL_TEXT=The manager's override number just entered is not defined.
 - B059_HANDLING_CLASS_KEY=FatalErrorHandler
 - B059_ERROR_CODE=INVALID_MANAGER_OVERRIDE_NUMBER
 - B059_EXTENDED_ERROR_CODE=NONE
- B060 Change Amount Limit Check
 - B060_ADDITIONAL_TEXT=The change limit for this tender type has been exceeded.
 - B060_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B060_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B060_EXTENDED_ERROR_CODE=CHANGE_AMOUNT_LIMIT
- B061 Item Entries Require Override
 - B061_ADDITIONAL_TEXT=Item or discount entries made during the transaction exceed user-defined limits and an override is required before payment for the transaction can be accepted. The entries which are being overridden are marked on the receipt tape.
 - B061_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B061_ERROR_CODE=MANAGER_OVERRIDE_REQUIRED
 - B061_EXTENDED_ERROR_CODE=NONE
- B062 Keyed Price Limit Check
 - B062_ADDITIONAL_TEXT=The price keyed for this item is too small for the department or too different from the price on the file.
 - B062_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B062_ERROR_CODE=INVALID_ARGUMENT
 - B062_EXTENDED_ERROR_CODE=INVALID_PRICE
- B063 Tender Amount Limit Check
 - B063_ADDITIONAL_TEXT=Tender amount exceeds the limit for this tender type.
 - B063_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B063_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B063_EXTENDED_ERROR_CODE=TENDER_AMOUNT_LIMIT
- B064 Coupon Must Match Previous
 - B064_ADDITIONAL_TEXT=No item in the order matches the coupon, or coupons have already been taken for all of the matching items.
 - B064_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B064_ERROR_CODE=NO_ITEM_MATCH_FOR_COUPON
 - B064_EXTENDED_ERROR_CODE=NONE
- B065 Coupon Value Exceeds Item Value
 - B065_ADDITIONAL_TEXT=The coupon matches items sold in the transaction but exceeds the value of any of those items, perhaps because of coupon value multiplication.
 - B065_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B065_ERROR_CODE=COUPON_VALUE_EXCEEDS_ITEM_VALUE
 - B065_EXTENDED_ERROR_CODE=NONE
- B066 Return Coupon Before Item Cancel
 - B066_ADDITIONAL_TEXT=A coupon which has been tendered against the item being voided must be voided before the item.
 - B066_HANDLING_CLASS_KEY=FatalErrorHandler
 - B066_ERROR_CODE=RETURN_COUPON_BEFORE_ITEM_VOID
 - B066_EXTENDED_ERROR_CODE=NONE
- B067 Cancel Must Match Previous Entry

- B067_ADDITIONAL_TEXT=A cancelation of an item, discount, or tender must match a previous entry to be accepted.
 - B067_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B067_ERROR_CODE=VOID_MUST_MATCH_PREVIOUS
 - B067_EXTENDED_ERROR_CODE=NONE
- B068 Full Key Sequence Required
 - B068_ADDITIONAL_TEXT=A Void, Enter, Enter key sequence cannot be used to void or repeat the previous entry because it was a negative entry or it included a weight, quantity, or manager's override.
 - B068_HANDLING_CLASS_KEY=FatalErrorHandler
 - B068_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B068_EXTENDED_ERROR_CODE=NONE
- B069 Tenders/Coupons Must be Returned
 - B069_ADDITIONAL_TEXT=A transaction has been voided after coupons or other tenders have been accepted.
 - B069_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B069_ERROR_CODE=NONE
 - B069_EXTENDED_ERROR_CODE=NONE
- B070 New Password Accepted
 - B070_ADDITIONAL_TEXT=A new password has been entered at sign-on and has been accepted.
 - B070_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B070_ERROR_CODE=NONE
 - B070_EXTENDED_ERROR_CODE=NONE
- B071 Password Must Match Previous Entry
 - B071_ADDITIONAL_TEXT=The new password entered at sign-on was not entered twice identically as required for acceptance.
 - B071_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B071_ERROR_CODE=NONE
 - B071_EXTENDED_ERROR_CODE=NONE
- B072 Password Needed ** Terminal Secured **
 - B072_ADDITIONAL_TEXT=To exit the **Terminal Secured** state, you must enter the password of the operator who secured the terminal.
 - B072_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B072_ERROR_CODE=INVALID_ARGUMENT
 - B072_EXTENDED_ERROR_CODE=PASSWORD_REQUIRED
- B073 Not Available in Standalone
 - B073_ADDITIONAL_TEXT=No externally verified tenders are accepted in standalone mode and the only non-sales procedures available are Tender Cashing and Tender Exchange.
 - B073_HANDLING_CLASS_KEY=FatalErrorHandler
 - B073_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B073_EXTENDED_ERROR_CODE=NOT_ALLOWED_OFFLINE
- B074 Terminal has Been Transferred
 - B074_ADDITIONAL_TEXT=After a terminal has been transferred, it must be re-initialized before it can be used for another sales transaction. A normal reload of the terminal or the running of diagnostic routines will suffice to re-initialize the terminal.
 - B074_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B074_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B074_EXTENDED_ERROR_CODE=NONE
- B075 Check Procedure Number
 - B075_ADDITIONAL_TEXT=The procedure number keyed is not defined.
 - B075_HANDLING_CLASS_KEY=FatalErrorHandler
 - B075_ERROR_CODE=INVALID_ARGUMENT
 - B075_EXTENDED_ERROR_CODE=INVALID_TRANSACTION_TYPE

- B076 Authorization Required
 - B076_ADDITIONAL_TEXT=The operator is not authorized for the attempted procedure or is not authorized to perform a discount, refund, or miscellaneous transaction payout.
 - B076_HANDLING_CLASS_KEY=FatalErrorHelper
 - B076_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B076_EXTENDED_ERROR_CODE=NOT_AUTHORIZED
- B077 Check Password
 - B077_ADDITIONAL_TEXT=The password entered for a sign-on or special sign-on is incorrect.
 - B077_HANDLING_CLASS_KEY=FatalErrorHelper
 - B077_ERROR_CODE=INVALID_ARGUMENT
 - B077_EXTENDED_ERROR_CODE=INVALID_PASSWORD
- B078 Check Operator Number
 - B078_ADDITIONAL_TEXT=The operator number entered with a sign-on is invalid.
 - B078_HANDLING_CLASS_KEY=FatalErrorHelper
 - B078_ERROR_CODE=INVALID_ARGUMENT
 - B078_EXTENDED_ERROR_CODE=INVALID_ID
- B079 Operator Still Active on Terminal Number
 - B079_ADDITIONAL_TEXT=An operator can sign on to one terminal at a time and this operator is already signed on at the terminal number shown in the guidance message.
 - B079_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B079_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B079_EXTENDED_ERROR_CODE=OPERATOR_ALREADY_ACTIVE
- B080 Key Operator, /, Password, Sign-on
 - B080_ADDITIONAL_TEXT=An invalid key sequence has been used on a terminal which is not yet signed on.
 - B080_HANDLING_CLASS_KEY=FatalErrorHelper
 - B080_ERROR_CODE=CONFIG_ERROR
 - B080_EXTENDED_ERROR_CODE=INVALID_KEY_SEQUENCE
- B081 Closing Period Wait and Retry
 - B081_ADDITIONAL_TEXT=For a brief period when the close of a reporting period is selected at the manager's terminal, other terminals are prohibited from signing on or off. A terminal might also be prohibited from starting a new transaction.
 - B081_HANDLING_CLASS_KEY=FatalErrorHelper
 - B081_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B081_EXTENDED_ERROR_CODE=CLOSING_ACCOUNTING_PERIOD
- B082 Pickup Needed Soon
 - B082_ADDITIONAL_TEXT=The first cash drawer limit has been exceeded and this guidance will be given at the end of every transaction until a pickup is made.
 - B082_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B082_ERROR_CODE=NONE
 - B082_EXTENDED_ERROR_CODE=NONE
- B083 Pickup Needed Now
 - B083_ADDITIONAL_TEXT=The second cash drawer limit has been exceeded and no further transactions can be started at this terminal until enough cash is picked up to bring the terminal below this limit.
 - B083_HANDLING_CLASS_KEY=FatalErrorHelper
 - B083_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B083_EXTENDED_ERROR_CODE=PICKUP_NEEDED
- B084 Not Available While Training

- B084_ADDITIONAL_TEXT=A sign-on to tender listing, operator training, terminal monitor, or terminal transfer is prohibited when in training mode.
 - B084_HANDLING_CLASS_KEY=FatalErrorHelper
 - B084_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B084_EXTENDED_ERROR_CODE=NOT_ALLOWED_TRAINING
- B085 Till Exchange Required
 - B085_ADDITIONAL_TEXT=A till exchange is required as the first transaction of every new reporting period.
 - B085_HANDLING_CLASS_KEY=FatalErrorHelper
 - B085_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B085_EXTENDED_ERROR_CODE=TILL_EXCHANGE_NEEDED
- B086 Monitor Already Active on Terminal Number
 - B086_ADDITIONAL_TEXT=The terminal monitor function may be run at one terminal in the store at a time. This includes the manager's terminal. The procedure is already active at the terminal address shown in the guidance. If monitor is active at the controller window, then the terminal number displayed is 999.
 - B086_HANDLING_CLASS_KEY=FatalErrorHelper
 - B086_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B086_EXTENDED_ERROR_CODE=MONITOR_ALREADY_ACTIVE
- B087 Terminal Must Match Previous Entry
 - B087_ADDITIONAL_TEXT=The terminal number to transfer was not entered twice identically as is required for acceptance.
 - B087_HANDLING_CLASS_KEY=FatalErrorHelper
 - B087_ERROR_CODE=INVALID_ARGUMENT
 - B087_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- B088 Cash Change Owed to Customer
 - B088_ADDITIONAL_TEXT=A transaction containing an approved EFT tender has been voided or a tender exchange has been made to correct an invalid tender entry and, as a result, cash change is owed to the customer.
 - B088_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B088_ERROR_CODE=NONE
 - B088_EXTENDED_ERROR_CODE=NONE
- B089 Cash Change Needed From Customer
 - B089_ADDITIONAL_TEXT=A tender exchange has been made to correct and invalid tender entry and, as a result, cash change which may previously have been given to the customer needs to be returned to the till.
 - B089_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B089_ERROR_CODE=NONE
 - B089_EXTENDED_ERROR_CODE=NONE
- B090 Check Tender Type
 - B090_ADDITIONAL_TEXT=A tender key has been used which is invalid for this function for one of the following reason: 1) Tender cashing cannot be used with cash and food stamps. 2) Tender listing is only valid for selected tender types. 3) Tender exchange is not allowed between selected tender types.
 - B090_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B090_ERROR_CODE=INVALID_ARGUMENT
 - B090_EXTENDED_ERROR_CODE=INVALID_TENDER_TYPE
- B091 Model Item Not Found
 - B091_ADDITIONAL_TEXT=An add of an item record has been attempted with the price change procedure using a model item record for default values and the model item is not on the file.
 - B091_HANDLING_CLASS_KEY=FatalErrorHelper
 - B091_ERROR_CODE=ITEM_NOT_FOUND
 - B091_EXTENDED_ERROR_CODE=NONE

- B092 Invalid Fee Amount
 - B092_ADDITIONAL_TEXT=Only selected fee amounts can be entered with any tender type during a tender exchange procedure.
 - B092_HANDLING_CLASS_KEY=FatalErrorHelper
 - B092_ERROR_CODE=INVALID_ARGUMENT
 - B092_EXTENDED_ERROR_CODE=INVALID_FEE_AMOUNT
- B093 Terminal Not a Master
 - B093_ADDITIONAL_TEXT=TPL can only be run from a designated master terminal. This terminal is not configured as a master terminal.
 - B093_HANDLING_CLASS_KEY=FatalErrorHelper
 - B093_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B093_EXTENDED_ERROR_CODE=NONE
- B094 Procedure Not Available
 - B094_ADDITIONAL_TEXT=A procedure has been attempted that is not available in the current transaction for one of the following reasons: 1) No-sale till exchange, tender verification, and till report are not allowed during a customer checkout transaction. 2) No help information is available in a customer checkout transaction. 3) A sign-off is not allowed within a checkout or non-sales transaction except for within a tender listing, terminal monitor, or terminal transfer transaction. 4) A sign-off is not allowed if the terminal experiences a failure in its write to totals retention.
 - B094_HANDLING_CLASS_KEY=FatalErrorHelper
 - B094_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B094_EXTENDED_ERROR_CODE=NONE
- B095 Item Change Alters Mix and Match
 - B095_ADDITIONAL_TEXT=A price or department number change has been made which alters the item set that can be mixed and matched with this item.
 - B095_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B095_ERROR_CODE=NONE
 - B095_EXTENDED_ERROR_CODE=NONE
- B096 Check Terminal Number
 - B096_ADDITIONAL_TEXT=The terminal number entered to be monitored or transferred is not active within the system.
 - B096_HANDLING_CLASS_KEY=FatalErrorHelper
 - B096_ERROR_CODE=INVALID_ARGUMENT
 - B096_EXTENDED_ERROR_CODE=INVALID_TERMINAL_NUMBER
- B097 Verification Timeout
 - B097_ADDITIONAL_TEXT=The terminal has not received a response for an external tender verification request within the selected time.
 - B097_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B097_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B097_EXTENDED_ERROR_CODE=VERIFICATION_TIMEOUT
- B098 Check PIN Pad
 - B098_ADDITIONAL_TEXT=A personal identification number is needed for external tender processing and cannot be obtained from the MSR/PIN device.
 - B098_HANDLING_CLASS_KEY=FatalErrorHelper
 - B098_ERROR_CODE=PIN_PAD_ERROR
 - B098_EXTENDED_ERROR_CODE=PIN_COULD_NOT_BE_OBTAINED
- B099 Use Another Tender
 - B099_ADDITIONAL_TEXT=The use of externally verified tenders has been temporarily suspended by the store manager until problems with this function can be corrected, or more than six externally verified tenders have been entered in a single transaction.
 - B099_HANDLING_CLASS_KEY=FatalErrorHelper
 - B099_ERROR_CODE=PROCEDURE_NOT_ALLOWED

- B099_EXTENDED_ERROR_CODE=EXTERNAL_TENDER_AUTHORIZATION_SUSPENDED
- B100 Card Has Expired
 - B100_ADDITIONAL_TEXT=The expiration date for the customer card has passed.
 - B100_HANDLING_CLASS_KEY=FatalErrorHelper
 - B100_ERROR_CODE=LOYALTY_CARD_EXPIRED
 - B100_EXTENDED_ERROR_CODE=NONE
- B106 Amount Must Balance
 - B106_ADDITIONAL_TEXT=Amount cannot be altered for EFT Refund tender.
 - B106_HANDLING_CLASS_KEY=FatalErrorHelper
 - B106_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B106_EXTENDED_ERROR_CODE=CUSTOMER_ALTERED_AMOUNT_AT_PINPAD
- B107 Age Check Needed (User Defined)
 - B107_ADDITIONAL_TEXT=Age Check Needed.
 - B107_HANDLING_CLASS_KEY=AgeRestrictedHelper
 - B107_ERROR_CODE=AGE_RESTRICTED_ITEM
 - B107_EXTENDED_ERROR_CODE=NONE
- B108 User Defined
 - B108_ADDITIONAL_TEXT=This message is reserved for the customer to use with external tender verification and/or user exit modifications.
 - B108_HANDLING_CLASS_KEY=DefaultErrorHelper
 - B108_ERROR_CODE=NONE
 - B108_EXTENDED_ERROR_CODE=NONE
- B109 User Defined
 - B109_ADDITIONAL_TEXT=This message is reserved for the customer to use with external tender verification and/or user exit modifications.
 - B109_HANDLING_CLASS_KEY=DefaultErrorHelper
 - B109_ERROR_CODE=NONE
 - B109_EXTENDED_ERROR_CODE=NONE
- B110 Invalid Display Requested
 - B110_ADDITIONAL_TEXT=A coding error has occurred, which led to the request of an invalid display message.
 - B110_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B110_ERROR_CODE=NONE
 - B110_EXTENDED_ERROR_CODE=NONE
- B111 Invalid Print Requested
 - B111_ADDITIONAL_TEXT=A coding error has occurred, which led to the request of an invalid print message.
 - B111_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B111_ERROR_CODE=NONE
 - B111_EXTENDED_ERROR_CODE=NONE
- B112 Tender Must Equal Balance Due
 - B112_ADDITIONAL_TEXT=The transaction is too large and only one more tender will be allowed at this time.
 - B112_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B112_ERROR_CODE=NONE
 - B112_EXTENDED_ERROR_CODE=NONE
- B113 Remove Franked Tender
 - B113_ADDITIONAL_TEXT=A print line needs to be printed on the customer receipt but cannot because a document is in the document insert station.
 - B113_HANDLING_CLASS_KEY=SAErrorHelperB113
 - B113_ERROR_CODE=NONE
 - B113_EXTENDED_ERROR_CODE=NONE

- B114 Item Concurrently Updated
 - B114_ADDITIONAL_TEXT= An item which has just been altered using the Price Verify/Change Procedure has been altered by some other process at the same time. The item will be updated as defined at this terminal and therefore the changes made by the other process may be lost.
 - B114_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B114_ERROR_CODE=NONE
 - B114_EXTENDED_ERROR_CODE=NONE
- B115 Check Batch Number
 - B115_ADDITIONAL_TEXT=A delayed maintenance batch number entered for an item record change transaction is invalid or has been overused.
 - B115_HANDLING_CLASS_KEY=FatalErrorHandler
 - B115_ERROR_CODE=INVALID_ARGUMENT
 - B115_EXTENDED_ERROR_CODE=INVALID_BATCH_NUMBER
- B116 Terminal must close
 - B116_ADDITIONAL_TEXT=The terminal has been forced to sign-off by a process at the manager's terminal.
 - B116_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B116_ERROR_CODE=NONE
 - B116_EXTENDED_ERROR_CODE=NONE
- B117 Please End Transaction
 - B117_ADDITIONAL_TEXT=The transaction is approaching the upper limit on the number of items allowed in a transaction.
 - B117_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B117_ERROR_CODE=NONE
 - B117_EXTENDED_ERROR_CODE=NONE
- B118 No More Items Allowed
 - B118_ADDITIONAL_TEXT=The transaction is too large and must be ended. Only tender entries will be accepted at this time.
 - B118_HANDLING_CLASS_KEY=FatalErrorHandler
 - B118_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B118_EXTENDED_ERROR_CODE=MAX_NUMBER_OF_ITEMS
- B119 Customer Altered Tender Amount
 - B119_ADDITIONAL_TEXT=The customer has altered the tender amount at the MSR/PIN device so that the amount no longer matches the balance due or the amount entered by the operator.
 - B119_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B119_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B119_EXTENDED_ERROR_CODE=CUSTOMER_ALTERED_AMOUNT_AT_PINPAD
- B120 Payment Type Mismatch
 - B120_ADDITIONAL_TEXT=The customer has selected a payment type on the MSR/PIN device which does not match the tender type entered at the terminal.
 - B120_HANDLING_CLASS_KEY=FatalErrorHandler
 - B120_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B120_EXTENDED_ERROR_CODE=CUSTOMER_CHOSE_ANOTHER_TENDER_AT_PINPAD
- B121 Enter Price Per Pound or Press Clear
 - B121_ADDITIONAL_TEXT=The scale requires the price per pound of the item.
 - B121_HANDLING_CLASS_KEY=FatalErrorHandler
 - B121_ERROR_CODE=SCALE_ERROR
 - B121_EXTENDED_ERROR_CODE=PRICE_PER_POUND_REQUIRED
- B122 Enter Price Per kilogram or Press Clear

- B122_ADDITIONAL_TEXT=The scale requires the price per kilogram of the item.
 - B122_HANDLING_CLASS_KEY=FatalErrorHandler
 - B122_ERROR_CODE=SCALE_ERROR
 - B122_EXTENDED_ERROR_CODE=PRICE_PER_KILOGRAM_REQUIRED
- B123 Weight key Required
 - B123_ADDITIONAL_TEXT=The scale requires that the Weight key be pressed.
 - B123_HANDLING_CLASS_KEY=DefaultErrorHandler
 - B123_ERROR_CODE=INVALID_ARGUMENT
 - B123_EXTENDED_ERROR_CODE=ITEM_WEIGHT_REQUIRED
- B124 Replace Item or Press Clear
 - B124_ADDITIONAL_TEXT=The scale requires that the operator replace the item on the scale or press Clear.
 - B124_HANDLING_CLASS_KEY=FatalErrorHandler
 - B124_ERROR_CODE=SCALE_ERROR
 - B124_EXTENDED_ERROR_CODE=REPLACE_ITEM
- B125 Weight Item or Press Clear
 - B125_ADDITIONAL_TEXT=The scale requires that the operator weigh the item on the scale or press Clear.
 - B125_HANDLING_CLASS_KEY=FatalErrorHandler
 - B125_ERROR_CODE=SCALE_ERROR
 - B125_EXTENDED_ERROR_CODE=WEIGH_ITEM
- B126 Check Price of Item
 - B126_ADDITIONAL_TEXT=The scale requires that the operator check the price of the item.
 - B126_HANDLING_CLASS_KEY=FatalErrorHandler
 - B126_ERROR_CODE=SCALE_ERROR
 - B126_EXTENDED_ERROR_CODE=CHECK_ITEM_PRICE
- B127 Item Cannot be Weighed
 - B127_ADDITIONAL_TEXT=The scale cannot weigh the item.
 - B127_HANDLING_CLASS_KEY=FatalErrorHandler
 - B127_ERROR_CODE=SCALE_ERROR
 - B127_EXTENDED_ERROR_CODE=ITEM_CANNOT_BE_WEIGHED
- B128 Check Weight Personalization
 - B128_ADDITIONAL_TEXT=The scale has encountered an error or inconsistency in the Weight Personalization options.
 - B128_HANDLING_CLASS_KEY=FatalErrorHandler
 - B128_ERROR_CODE=SCALE_ERROR
 - B128_EXTENDED_ERROR_CODE=CHECK_WEIGHT_PERSONALIZATION
- B129 Check Override Price
 - B129_ADDITIONAL_TEXT=The scale has encountered an error or inconsistency in the override price.
 - B129_HANDLING_CLASS_KEY=FatalErrorHandler
 - B129_ERROR_CODE=SCALE_ERROR
 - B129_EXTENDED_ERROR_CODE=CHECK_OVERRIDE_PRICE
- B130 Item Not on Scale Correctly
 - B130_ADDITIONAL_TEXT=The scale cannot weigh the item because the item is not placed properly on the scale.
 - B130_HANDLING_CLASS_KEY=FatalErrorHandler
 - B130_ERROR_CODE=SCALE_ERROR
 - B130_EXTENDED_ERROR_CODE=ITEM_NOT_ON_SCALE_CORRECTLY
- B131 Check Scale Configuration

- B131_ADDITIONAL_TEXT=The scale has encountered an error in its configuration.
 - B131_HANDLING_CLASS_KEY=FatalErrorHelper
 - B131_ERROR_CODE=SCALE_ERROR
 - B131_EXTENDED_ERROR_CODE=CHECK_CONFIGURATION
- B132 Previous Item Still on Scale
 - B132_ADDITIONAL_TEXT=The scale still holds the item previously weighed.
 - B132_HANDLING_CLASS_KEY=FatalErrorHelper
 - B132_ERROR_CODE=SCALE_ERROR
 - B132_EXTENDED_ERROR_CODE=PREVIOUS_ITEM_ON_SCALE
- B133 Check Scale Connection
 - B133_ADDITIONAL_TEXT=The scale has encountered a problem with its connection to the terminal.
 - B133_HANDLING_CLASS_KEY=FatalErrorHelper
 - B133_ERROR_CODE=SCALE_ERROR
 - B133_EXTENDED_ERROR_CODE=DEVICE_OFFLINE
- B134 Press Enter to Sell or Clear
 - B134_ADDITIONAL_TEXT=The scale requires that you press the Enter or Clear key.
 - B134_HANDLING_CLASS_KEY=DefaultErrorHelper
 - B134_ERROR_CODE=SCALE_ERROR
 - B134_EXTENDED_ERROR_CODE=NONE
- B135 Check Keyed Weight
 - B135_ADDITIONAL_TEXT=The scale has encountered a problem with the keyed weight.
 - B135_HANDLING_CLASS_KEY=FatalClearErrorHelper
 - B135_ERROR_CODE=INVALID_ARGUMENT
 - B135_EXTENDED_ERROR_CODE=INVALID_WEIGHT
- B429 No Message Text Found
 - B429_ADDITIONAL_TEXT=The application has detected an error in one of the fields in the Currency Definition file, EAMEXCHG.DAT, or in the ISO Country ID file, EAMEXISO.DAT, and has disabled the Multiple Currency Feature.
 - B429_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B429_ERROR_CODE=POS_APP_FAILURE
 - B429_EXTENDED_ERROR_CODE=INVALID_APPLICATION_DATA
- B503 Too Many EFT Tenders
 - B503_ADDITIONAL_TEXT=The number of EFT tenders allowed has been exceeded for this transaction. The application will accept five EFT tenders per transaction.
 - B503_HANDLING_CLASS_KEY=FatalErrorHelper
 - B503_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B503_EXTENDED_ERROR_CODE=NUMBER_OF_TENDERS_LIMIT
- B504 Use Another Tender
 - B504_ADDITIONAL_TEXT=Accepting EFT tender has not been activated, or you are offline from the controller.
 - B504_HANDLING_CLASS_KEY=FatalErrorHelper
 - B504_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B504_EXTENDED_ERROR_CODE=CREDIT_NOT_AVAILABLE
- B505 EFT Device is not Available
 - B505_ADDITIONAL_TEXT=The EFT device is not available due to program/parameters being loaded, or the device is not known to the terminal.
 - B505_HANDLING_CLASS_KEY=FatalErrorHelper
 - B505_ERROR_CODE=PIN_PAD_ERROR
 - B505_EXTENDED_ERROR_CODE=NONE

- B506 Invalid Tender Type
 - B506_ADDITIONAL_TEXT=The use of externally verified tenders and Electronic Funds Transfer is mutually exclusive.
 - B506_HANDLING_CLASS_KEY=FatalErrorHandler
 - B506_ERROR_CODE=INVALID_ARGUMENT
 - B506_EXTENDED_ERROR_CODE=INVALID_TENDER_TYPE
- B507 Reload Terminal Item File
 - B507_ADDITIONAL_TEXT=Changes to the controller Item Record File have been made through the Terminal Price Verification Procedure. These changes have caused the Terminal Maintenance Control File (EAMTMCFR) to exceed its threshold size.
 - B507_HANDLING_CLASS_KEY=FatalErrorHandler
 - B507_ERROR_CODE=NONE
 - B507_EXTENDED_ERROR_CODE=NONE
- B508 Invalid Date Entered
 - B508_ADDITIONAL_TEXT=The date entered for the expiration date is invalid.
 - B508_HANDLING_CLASS_KEY=FatalErrorHandler
 - B508_ERROR_CODE=INVALID_ARGUMENT
 - B508_EXTENDED_ERROR_CODE=INVALID_EXPIRY_DATE
- B509 Log is Full
 - B509_ADDITIONAL_TEXT=The Terminal Transaction Log (EAQMTLxxx) has used all space available in the terminal RAM disk. The transaction cannot be saved in terminal memory.
 - B509_HANDLING_CLASS_KEY=FatalErrorHandler
 - B509_ERROR_CODE=POS_APP_FAILURE
 - B509_EXTENDED_ERROR_CODE=FILE_IO_ERROR
- B511 Check the EFT Device
 - B511_ADDITIONAL_TEXT=The EFT device is not communicating with the terminal.
 - B511_HANDLING_CLASS_KEY=FatalErrorHandler
 - B511_ERROR_CODE=PIN_PAD_ERROR
 - B511_EXTENDED_ERROR_CODE=COMMUNICATION_ERROR
- B515 Zero EFT amount Not Allowed
 - B515_ADDITIONAL_TEXT=The amount tendered for an EFT transaction is 0.
 - B515_HANDLING_CLASS_KEY=FatalErrorHandler
 - B515_ERROR_CODE=INVALID_ARGUMENT
 - B515_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- B516 EFT Device is Loading
 - B516_ADDITIONAL_TEXT=EFT device is being loaded and is unavailable until the load is complete.
 - B516_HANDLING_CLASS_KEY=FatalErrorHandler
 - B516_ERROR_CODE=PIN_PAD_ERROR
 - B516_EXTENDED_ERROR_CODE=DEVICE_IS_LOADING
- B517 Change of Amount Failed
 - B517_ADDITIONAL_TEXT=The operator changed the tender amount as the customer pressed AMOUNT OK on the EFT device (alternate system only).
 - B517_HANDLING_CLASS_KEY=FatalErrorHandler
 - B517_ERROR_CODE=INVALID_ARGUMENT
 - B517_EXTENDED_ERROR_CODE=INVALID_AMOUNT
- B530 Response Code xx
 - B530_ADDITIONAL_TEXT=This is the response code from the servicer or controller for the EFT transaction. This message will be displayed if you have not personalized a message for this response code in the "EFT Servicer Detail" section.

- B530_HANDLING_CLASS_KEY=FatalErrorHandler
 - B530_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B530_EXTENDED_ERROR_CODE=NONE
- B531 Approved
 - B531_ADDITIONAL_TEXT=This is the default message for all approval responses to EFT transactions.
 - B531_HANDLING_CLASS_KEY=DefaultErrorHandler
 - B531_ERROR_CODE=NONE
 - B531_EXTENDED_ERROR_CODE=NONE
- B532 Not Authorized
 - B532_ADDITIONAL_TEXT=This is the default message for all rejected responses to EFT transactions.
 - B532_HANDLING_CLASS_KEY=FatalErrorHandler
 - B532_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B532_EXTENDED_ERROR_CODE=TENDER_NOT_AUTHORIZED
- B533 Re-enter Code
 - B533_ADDITIONAL_TEXT=The PIN number must be reentered by the customer.
 - B533_HANDLING_CLASS_KEY=AppDefaultErrorHandler
 - B533_ERROR_CODE=NONE
 - B533_EXTENDED_ERROR_CODE=NONE
- B534 Too Long in Stand-in
 - B534_ADDITIONAL_TEXT=The amount of time allowed for the store to process in stand-in mode has been exceeded.
 - B534_HANDLING_CLASS_KEY=FatalErrorHandler
 - B534_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B534_EXTENDED_ERROR_CODE=TOO_LONG_IN_STAND_IN
- B535 Stand-in Count Exceeded
 - B535_ADDITIONAL_TEXT=The number of transactions allowed by the store to process in stand-in mode has been exceeded.
 - B535_HANDLING_CLASS_KEY=FatalErrorHandler
 - B535_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B535_EXTENDED_ERROR_CODE=STAND_IN_COUNT_LIMIT
- B536 Stand-in Amount Exceeded
 - B536_ADDITIONAL_TEXT=The total amount of money to be accepted in stand-in mode has been exceeded.
 - B536_HANDLING_CLASS_KEY=FatalErrorHandler
 - B536_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B536_EXTENDED_ERROR_CODE=STAND_IN_AMOUNT_LIMIT
- B537 No Stand-in for This Servicer
 - B537_ADDITIONAL_TEXT=The servicer for this account does not allow the store to process EFT in stand-in mode.
 - B537_HANDLING_CLASS_KEY=FatalErrorHandler
 - B537_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B537_EXTENDED_ERROR_CODE=PAYMENT_SYSTEM_OFFLINE
- B538 Risk 1
 - B538_ADDITIONAL_TEXT=The customer card is in the negative tender verification file and cannot be accepted. Follow store policy.
 - B538_HANDLING_CLASS_KEY=FatalErrorHandler
 - B538_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B538_EXTENDED_ERROR_CODE=RISK_1
- B539 Floor Limit Exceeded
 - B539_ADDITIONAL_TEXT=The limit set by the store for the amount that can be processed without requiring a manager's approval.
 - B539_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B539_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED

- B539_EXTENDED_ERROR_CODE=TENDER_FLOOR_LIMIT
- B540 Card has Expired
 - B540_ADDITIONAL_TEXT=The customer's card has expired.
 - B540_HANDLING_CLASS_KEY=FatalErrorHandler
 - B540_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B540_EXTENDED_ERROR_CODE=CARD_EXPIRED
- B541 Servicer not Known
 - B541_ADDITIONAL_TEXT=The servicer for this transaction is unknown to the host.
 - B541_HANDLING_CLASS_KEY=FatalErrorHandler
 - B541_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B541_EXTENDED_ERROR_CODE=UNKNOWN_SERVICER
- B542 Servicer Closed
 - B542_ADDITIONAL_TEXT=The servicer defined to the system is closed. Servicer hours are defined in the "Servicer Detail" section of chapter 4 of the EFT guide.
 - B542_HANDLING_CLASS_KEY=FatalErrorHandler
 - B542_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B542_EXTENDED_ERROR_CODE=SERVICER_CLOSED
- B543 Card Type Not Known
 - B543_ADDITIONAL_TEXT=The customer's bank identification data is not in the BIN file.
 - B543_HANDLING_CLASS_KEY=FatalErrorHandler
 - B543_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B543_EXTENDED_ERROR_CODE=CARD_TYPE_UNKNOWN
- B560 Response xxx
 - B560_ADDITIONAL_TEXT=This is the response code from the servicer or controller for the EFT transaction that is displayed on the EFT device. This message will be displayed if you have not personalized a message for this response code in the "EFT Servicer Detail" section of the EFT guide.
 - B560_HANDLING_CLASS_KEY=FatalErrorHandler
 - B560_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B560_EXTENDED_ERROR_CODE=NONE
- B561 Approved
 - B561_ADDITIONAL_TEXT=These are the default messages for all approval responses to EFT transactions that are displayed on the EFT device.
 - B561_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B561_ERROR_CODE=NONE
 - B561_EXTENDED_ERROR_CODE=NONE
- B562 Not Authorized
 - B562_ADDITIONAL_TEXT=The default messages for all rejected responses to EFT transactions that are displayed on the EFT device.
 - B562_HANDLING_CLASS_KEY=FatalErrorHandler
 - B562_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B562_EXTENDED_ERROR_CODE=TENDER_NOT_AUTHORIZED
- B563 Re-enter Code
 - B563_ADDITIONAL_TEXT=The PIN number must be re-entered by the customer.
 - B563_HANDLING_CLASS_KEY=FatalErrorHandler
 - B563_ERROR_CODE=NONE
 - B563_EXTENDED_ERROR_CODE=NONE
- B600 Points Entry Requires Customer Number
 - B600_ADDITIONAL_TEXT=A Preferred Customer ID is needed to award bonus points or to redeem points with discount coupons.
 - B600_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B600_ERROR_CODE=TENDER_NOT_ACCEPTED

- B600_EXTENDED_ERROR_CODE=LOYALTY_NUMBER_REQUIRED
- B601 Home Store Must Redeem Points
 - B601_ADDITIONAL_TEXT=A customer can redeem points for discounts only in his designated home store.
 - B601_HANDLING_CLASS_KEY=FatalErrorHandler
 - B601_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B601_EXTENDED_ERROR_CODE=HOME_STORE_MUST_REDEEM_LOYALTY_POINTS
- B602 More Points Needed for Discount
 - B602_ADDITIONAL_TEXT=The Preferred Customer has not accumulated enough points to qualify for the discount.
 - B602_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B602_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B602_EXTENDED_ERROR_CODE=MORE_LOYALTY_POINTS_NEEDED
- B603 No Available Drawer
 - B603_ADDITIONAL_TEXT=The cash drawer cannot be reserved at sign-off because the other drawer is already reserved by the operator whose number is displayed or only one cash drawer is functional.
 - B603_HANDLING_CLASS_KEY=FatalErrorHandler
 - B603_ERROR_CODE=CASH_DRAWER_ERROR
 - B603_EXTENDED_ERROR_CODE=NONE_AVAILABLE
- B604 Check Keyed Customer Number
 - B604_ADDITIONAL_TEXT=An override is required to key enter a customer number.
 - B604_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B604_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B604_EXTENDED_ERROR_CODE=NONE
- B605 Check Customer Number
 - B605_ADDITIONAL_TEXT=The customer number just entered is invalid or unexpected.
 - B605_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B605_ERROR_CODE=INVALID_ARGUMENT
 - B605_EXTENDED_ERROR_CODE=INVALID_LOYALTY_NUMBER
- B606 Redemption Must be Positive
 - B606_ADDITIONAL_TEXT=More redemption coupons have been voided than were taken in an order.
 - B606_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B606_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B606_EXTENDED_ERROR_CODE=NONE
- B607 Bonus Point Limit Check
 - B607_ADDITIONAL_TEXT=The number of bonus points awarded to the Preferred Customer in this order exceeds the store limit.
 - B607_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B607_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B607_EXTENDED_ERROR_CODE=LOYALTY_POINTS_LIMIT
- B608 Ask for Card Clear to Continue
 - B608_ADDITIONAL_TEXT=At total time, this guidance is just a reminder to ask the customer for his ID card. In response to a coupon entry, this guidance limits the use of the coupon to Preferred Customers.
 - B608_HANDLING_CLASS_KEY=SAErrorHandlerB608
 - B608_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B608_EXTENDED_ERROR_CODE=LOYALTY_NUMBER_REQUIRED
- B609 Checked Keyed Coupon Number
 - B609_ADDITIONAL_TEXT= An override is required to key a preferred coupon.
 - B609_HANDLING_CLASS_KEY=OverrideErrorHandler

- B609_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B609_EXTENDED_ERROR_CODE=NONE
- B610 Limited Number of Coupons Per Order
 - B610_ADDITIONAL_TEXT=Too many of this same coupon have been used in the order.
 - B610_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B610_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B610_EXTENDED_ERROR_CODE=NUMBER_OF_COUPONS_LIMIT
- B611 Coupon has Expired
 - B611_ADDITIONAL_TEXT=The expiration date for the coupon has passed.
 - B611_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B611_ERROR_CODE=COUPON_EXPIRED
 - B611_EXTENDED_ERROR_CODE=NONE
- B612 Minimum Sale Not Satisfied
 - B612_ADDITIONAL_TEXT=The minimum purchase requirements for this coupon have not been met in this order.
 - B612_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B612_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B612_EXTENDED_ERROR_CODE=MINIMUM_SALE_NOT_SATISFIED
- B613 Please Write XX on Coupon
 - B613_ADDITIONAL_TEXT=The value of the item given away with the "free item" coupon must be written on the coupon.
 - B613_HANDLING_CLASS_KEY=InformationalErrorHandler
 - B613_ERROR_CODE=NONE
 - B613_EXTENDED_ERROR_CODE=NONE
- B671 Function Not Available
 - B671_ADDITIONAL_TEXT=Suspend/Retrieve has been selected but Store or Terminal options have disabled the terminal from doing any suspensions or retrievals.
 - B671_HANDLING_CLASS_KEY=FatalErrorHandler
 - B671_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B671_EXTENDED_ERROR_CODE=NONE
- B673 Number of Suspends Exceeded
 - B673_ADDITIONAL_TEXT=The maximum number of suspended transactions has already been reached.
 - B673_HANDLING_CLASS_KEY=OverrideErrorHandler
 - B673_ERROR_CODE=APPLICATION_LIMIT_EXCEEDED
 - B673_EXTENDED_ERROR_CODE=SUSPENDED_TRANSACTION_LIMIT
- B676 Active Suspends, Clear or Continue
 - B676_ADDITIONAL_TEXT=The operator has active suspends when signing off.
 - B676_HANDLING_CLASS_KEY=SAErrorHandlerB676
 - B676_ERROR_CODE=NONE
 - B676_EXTENDED_ERROR_CODE=NONE
- B677 Retry Where Suspended
 - B677_ADDITIONAL_TEXT=The transaction must be retrieved at the terminal that suspended it.
 - B677_HANDLING_CLASS_KEY=FatalErrorHandler
 - B677_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B677_EXTENDED_ERROR_CODE=RETRIEVE_TRANSACTION_WHERE_SUSPENDED
- B678 Unable to Retrieve Data
 - B678_ADDITIONAL_TEXT=The transaction cannot be retrieved. The transaction has not been suspended successfully.
 - B678_HANDLING_CLASS_KEY=FatalErrorHandler
 - B678_ERROR_CODE=PROCEDURE_NOT_ALLOWED

- B678_EXTENDED_ERROR_CODE=UNABLE_TO_RETRIEVE_TRANSACTION
- B680 Transaction Retrieved Already
 - B680_ADDITIONAL_TEXT=The transaction has already been retrieved or was not suspended successfully.
 - B680_HANDLING_CLASS_KEY=FatalErrorHelper
 - B680_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B680_EXTENDED_ERROR_CODE=TRANSACTION_ALREADY_RETRIEVED
- B683 Same Operator Must Retrieve
 - B683_ADDITIONAL_TEXT=The operator that suspended the transaction must retrieve it.
 - B683_HANDLING_CLASS_KEY=FatalErrorHelper
 - B683_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B683_EXTENDED_ERROR_CODE=SAME_OPERATOR_MUST_RETRIEVE_TRANSACTION
- B684 Force Retrieve Required
 - B684_ADDITIONAL_TEXT=The manager is required to retrieve this transaction.
 - B684_HANDLING_CLASS_KEY=OverrideErrorHelper
 - B684_ERROR_CODE=PROCEDURE_NOT_ALLOWED
 - B684_EXTENDED_ERROR_CODE=NONE
- B685 Suspension Incomplete
 - B685_ADDITIONAL_TEXT=A problem has occurred during a suspension. The transaction has not been suspended successfully.
 - B685_HANDLING_CLASS_KEY=FatalErrorHelper
 - B685_ERROR_CODE=POS_APP_FAILURE
 - B685_EXTENDED_ERROR_CODE=NONE
- B686 Retrieval Incomplete
 - B686_ADDITIONAL_TEXT=A problem has occurred during a retrieval. The transaction has not been successfully retrieved.
 - B686_HANDLING_CLASS_KEY=FatalErrorHelper
 - B686_ERROR_CODE=POS_APP_FAILURE
 - B686_EXTENDED_ERROR_CODE=NONE
- B689 Invalid Tender
 - B689_ADDITIONAL_TEXT=The PIN pad must be present or this is not an EFT tender.
 - B689_HANDLING_CLASS_KEY=FatalErrorHelper
 - B689_ERROR_CODE=TENDER_NOT_ACCEPTED
 - B689_EXTENDED_ERROR_CODE=PIN_PAD_REQUIRED
- B870 Verify Signature
 - B870_ADDITIONAL_TEXT=Verify the signature on the check.
 - B870_HANDLING_CLASS_KEY=SignatureErrorHelper
 - B870_ERROR_CODE=VERIFY_SIGNATURE
 - B870_EXTENDED_ERROR_CODE=NONE
- B893 Printer Out of Paper, Printer Cover Open
 - B893_ADDITIONAL_TEXT=The printer customer receipt station is out of paper or the station cover has been opened.
 - B893_HANDLING_CLASS_KEY=PrinterPaperOutErrorHelper
 - B893_ERROR_CODE=PRINTER_ERROR
 - B893_EXTENDED_ERROR_CODE=NONE
- B894 Invalid Printer Command
 - B894_ADDITIONAL_TEXT=The printer has received a control that it does not recognize.
 - B894_HANDLING_CLASS_KEY=InformationalErrorHelper
 - B894_ERROR_CODE=POS_APP_FAILURE

- B894_EXTENDED_ERROR_CODE=UNRECOGNIZED_PRINT_CHARACTERS
- B895 Print Buffer Full
 - B895_ADDITIONAL_TEXT=The printer customer receipt station buffer is full.
 - B895_HANDLING_CLASS_KEY=FatalErrorHandler
 - B895_ERROR_CODE=PRINTER_ERROR
 - B895_EXTENDED_ERROR_CODE=BUFFER_FULL
- REMOVE RECEIPT FOR CUSTOMER (EAMEESDS.DAT Descriptor 20)
 - EFT_DESC20_ADDITIONAL_TEXT=Remove the receipt and hand it to the customer.
 - EFT_DESC20_HANDLING_CLASS_KEY=RemoveReceiptErrorHandler
 - EFT_DESC20_ERROR_CODE=PRINTER_ERROR
 - EFT_DESC20_EXTENDED_ERROR_CODE=NONE
- ENTER FOR DUPLICATE CLEAR TO BYPASS
 - 1100002_ADDITIONAL_TEXT=Press Enter to print a duplicate receipt or Clear to skip printing the duplicate(s).
 - 1100002_HANDLING_CLASS_KEY=DuplicateReceiptHelper
 - 1100002_ERROR_CODE=NONE
 - 1100002_EXTENDED_ERROR_CODE=NONE

nextFile=usererror – This property is used to tell AEF the name of the next property file in the bundle chain.

Function Code Bundle

The Function code bundle is used to map a logical key name to a function code. The files that makeup the Function code bundle are fcode.properties, appfcode.properties, and userfcode.properties

For each property, a key is provided followed by an equal sign and a value. These values must match the function code values for the base application.

Fcode.properties

The fcode.properties file includes the following properties.

- CLEAR=73

nextFile=appfcode - This property is used to tell AEF the name of the next property file in the bundle chain.

Appfcode.properties (ACE)

The appfcode.properties file includes the following properties.

- CASH=91
- CHECK=92
- CHECKVERIFY=90
- CREDIT=94
- CUSTOMER=89
- DATAENTRY=63
- DISCOUNT=62
- ENTER=80
- FOODSTAMP=93
- FSNOFS=76

- MFRCOUPON=65
- NOSALE=100
- OVERRIDE=79
- PRICE=74
- QTY=75
- REFUND=67
- SLASH=78
- SIGNON=61
- STORECOUPON=66
- SUSPENDRET=82
- TARE=71
- TAXNOTAX=77
- TOTAL=81
- VOID=70
- WEIGHT=72

nextFile=usercode - This property is used to tell AEF the name of the next property file in the bundle chain.

Appfcode.properties (GSA)

The appfcode.properties file includes the following properties.

- ALLOWCODE=67
- AMERICANEXPRESS=113
- CASH=69
- CHECK=101
- CLASS=83
- CLEAR=73
- CREDITPLANA=107
- CREDITPLANB=108
- CREDITPLANC=109
- CREDITPLAND=110
- DATAENTRY=84
- DEPARTMENT=79
- DEPARTMENT1=122
- DEPARTMENT2=123
- DEPARTMENT3=124
- DEPARTMENT4=125
- DEPARTMENT5=126
- DEPARTMENT6=127
- DEPARTMENT7=128
- DEPARTMENT8=129
- DEPARTMENT9=130
- DEPARTMENT10=131
- DEPARTMENT11=132
- DEPARTMENT12=133
- DEPARTMENT13=134
- DEPARTMENT14=135
- DEPARTMENT15=136
- DEPARTMENT16=137
- DINERSCLUB=114
- DISCOUNTCODE=64
- DUEBILL=103
- ENTER=95

- GIFTCERTIFICATE=102
- MASTERCARD=112
- MISCELLANEOUS=115
- MODIFYTICKET=74
- NONMDSECODE=65
- NONSALES=71
- NONTAX=80
- PAYMENT=68
- QTY=96
- RETURNCREDIT=62
- SIGNONOFF=76
- SKU=88
- SLASH=78
- STOCK=87
- STORECOUPON=105
- SUBTOTAL=66
- SUSPENDRET=77
- TAXCODE=63
- TOTAL=75
- TRAVELERSCHECK=104
- VENDORCOUPON=106
- VISA=111
- VOID=61

nextFile=userfcode - This property is used to tell AEF the name of the next property file in the bundle chain.

Appfcode.properties (SA)

The appfcode.properties file includes the following properties.

- CASH=91
- CHECK=92
- CHECKVERIFY=90
- CREDIT=94
- CUSTOMER=89
- DATAENTRY=63
- DISCOUNT=62
- ENTER=80
- FOODSTAMP=93
- FSNOFS=76
- MFRCOUPON=65
- NOSALE=100
- OVERRIDE=79
- PRICE=74
- QTY=75
- REFUND=67
- SLASH=78
- SIGNON=61
- STORECOUPON=66
- SUSPENDRET=82
- TARE=71
- TAXNOTAX=77
- TOTAL=81
- VOID=70

- WEIGHT=72

nextFile=userfcode - This property is used to tell AEF the name of the next property file in the bundle chain.

Internalization Bundle

The Internalization bundle is used to store any text that will be displayed to the end user of an AEF application. Therefore the text in the files that makeup this bundle must be translated into the local language. The files that makeup the Internalization bundle are i18ntext.properties, appi18ntext.properties, and useri18ntext.properties.

For each property, a key is provided followed by an equal sign and a value. The double quotes around the values are used to tell the NLS team what text to change. They will be removed by the I18nText.class when the text is read.

I18ntext.properties

The i18ntext.properties file includes the following properties.

- creatingSession="AEF: Creating session"
- creatingDataProvider="AEF: Creating POS DataProvider"
- creatingAutomation="AEF: Creating POS Automation"
- creatingHooks="AEF: Creating POS Device Hooks"
- creatingDeviceManager="AEF: Creating Device Manager"
- creatingJIOP="AEF: Creating IO Processor"
- loadingDeviceHandlers="AEF: Loading Device Handlers"
- creatingDeviceRegistry="AEF: Creating Device Registry"
- sessionJIOPReady="AEF: JIOP is ready"
- sessionReady="AEF: Session ready for input"
- sessionEnded="AEF: IO Processor terminated"

nextFile=appi18ntext - This property is used to tell AEF the name of the next property file in the bundle chain.

Appi18ntext.properties (ACE)

The appi18ntext.properties file includes the following properties.

nextFile=useri18ntext - This property is used to tell AEF the name of the next property file in the bundle chain.

Appi18ntext.properties (GSA)

The appi18ntext.properties file includes the following properties.

- managerOverrideLine1="Managers Key & Enter"
- managerOverrideLine2="or Clear to Cancel "

nextFile=useri18ntext - This property is used to tell AEF the name of the next property file in the bundle chain.

Appi18ntext.properties (SA)

The appi18ntext.properties file includes the following properties.

nextFile=useri18ntext - This property is used to tell AEF the name of the next property file in the bundle chain.

Key Sequence Bundle

The Key Sequence bundle is used to specify the key sequence required to do a particular POS operation on a terminal. The files that makeup the Key Sequence bundle are sequence.properties, appseq.properties, and userseq.properties

For each property, a key is provided followed by an equal sign and a value. If the value (or parts of the value) is surrounded by { }, then this text will be replaced by a variable at runtime. If the value (or parts of the value) is surrounded by < >, then the text will be replaced by the function code specified in the function code bundle for that key. If the value (or parts of the value) is surrounded by [\$], then the text will be replaced by the appropriate POSDataProvider property value at runtime. Any other text will be assumed to be part of the keying sequence and will be sent to the application without modification.

Sequence.properties

The sequence.properties file includes the following properties.

- clear-error=<CLEAR>
- total=<TOTAL>

nextFile=appseq - This property is used to tell AEF the name of the next property file in the bundle chain.

Appseq.properties (ACE)

The appsequence.properties file includes the following properties.

- accountNumber={0}<ENTER>
- alternateCustomer={0}<CUSTOMER>
- applyDelayedCoupons=<TOTAL>
- authorizationCode={0}<ENTER>
- birthdateEntry={0}<ENTER>
- cancelOverride=<CLEAR>
- cancelPrevious=<VOID><ENTER>
- cardID={0}<ENTER>
- creditAmount={0}<CREDIT>
- customer=<SLASH>{0}<ENTER>
- enterStandAlone=<SLASH><ENTER>
- expiryDate={0}<ENTER>
- id={0}<SLASH>
- item={0}<ENTER>
- itemQty={0}<QTY>
- itemPrice={0}<PRICE>
- itemDealPrice={0}<SLASH>{1}<PRICE>
- itemWeight={0}<WEIGHT>
- itemPriceQty={1}<PRICE>{2}<QTY>{0}<ENTER>
- itemPriceWeight={1}<PRICE>{2}<WEIGHT>{0}<ENTER>

- itemVoid=<VOID>{0}<ENTER>
- itemVoidPrice=<VOID>{1}<PRICE>{0}<ENTER>
- itemVoidQty=<VOID>{1}<QTY>{0}<ENTER>
- itemVoidWeight=<VOID>{1}<WEIGHT>{0}<ENTER>
- itemVoidPriceQty=<VOID>{1}<PRICE>{2}<QTY>{0}<ENTER>
- itemVoidPriceWeight=<VOID>{1}<PRICE>{2}<WEIGHT>{0}<ENTER>
- itemReturn=<REFUND>{0}<ENTER>
- itemReturnPrice=<REFUND>{1}<PRICE>{0}<ENTER>
- itemReturnQty=<REFUND>{1}<QTY>{0}<ENTER>
- itemReturnWeight=<REFUND>{1}<WEIGHT>{0}<ENTER>
- itemReturnPriceQty=<REFUND>{1}<PRICE>{2}<QTY>{0}<ENTER>
- itemReturnPriceWeight=<REFUND>{1}<PRICE>{2}<WEIGHT>{0}<ENTER>
- logoff1=<SIGNON>
- logoff2=<SIGNON>
- operatorOverride=<OVERRIDE><ENTER>
- password={0}<SIGNON>
- return=<REFUND>
- startTransaction=0<SIGNON>
- tenderVariety={0}<SLASH>
- voidTransaction=<VOID><TOTAL>
- managerOverride=<OVERRIDE>{0}<ENTER>
- total=<TOTAL>
- transDataEntry=<DATAENTRY>{0}<ENTER>
- suspendTransaction=<TOTAL><SUSPENDRET><TOTAL>
- retrieveTransaction=<SUSPENDRET>{0}<SLASH>{1}<ENTER>
- ignoreSuspended=<CLEAR>
- specialSignonToOverride=<SIGNON>
- trainingModeOn=7<SIGNON>
- verifySignature=<ENTER>
- void=<VOID>
- voucherNum={0}<ENTER>

nextFile=userseq - This property is used to tell AEF the name of the next property file in the bundle chain.

Appseq.properties (GSA)

The appsequence.properties file includes the following properties.

- account={0}<ENTER>
- ageRestrictBypass=<ENTER>
- amount={0}<ENTER>
- authorization={0}<ENTER>
- cancelPrevious=<VOID><ENTER>
- class={0}<CLASS>
- clear=<CLEAR>
- department={0}<DEPARTMENT>
- doubleClear=<CLEAR><CLEAR>
- enter=<ENTER>
- id={0}<SIGNONOFF>
- item={0}<ENTER>
- logoff1=<SIGNONOFF>
- logoff2=<ENTER>
- nontax=<NONTAX>
- nonmerchandise={0}<NONMDSECODE>

- originalSalesperson={0}<ENTER>
- password={0}<ENTER>
- price={0}<ENTER>
- quantity={0}<QTY>
- sku={0}<SKU>
- splitPrice={0}<SLASH>{1}<ENTER>
- startTransaction={0}<ENTER>
- stock={0}<STOCK>
- suspendTransaction={0}<SUSPENDRET>
- tender={0}{1}
- total=<TOTAL>
- transDataEntry={0}<DATAENTRY>
- retrieveTransaction={0}<SLASH>{1}<SUSPENDRET>
- return=<RETURNCREDIT>
- trainingModeOn=9<NONSEALES>
- void=<VOID>
- voidTransaction=<VOID><TOTAL>

nextFile=userseq - This property is used to tell AEF the name of the next property file in the bundle chain.

Appseq.properties (SA)

The appsequence.properties file includes the following properties.

- accountNumber={0}<ENTER>
- alternateCustomer={0}<CUSTOMER>
- applyDelayedCoupons=<TOTAL>
- birthdateEntry={0}<ENTER>
- cancelOverride=<CLEAR>
- cancelPrevious=<VOID><ENTER>
- creditAmount={0}<CREDIT>
- customer=<SLASH>{0}<ENTER>
- enterStandAlone=<SLASH><ENTER>
- expiryDate={0}<SLASH>
- id={0}<SLASH>
- ignoreSuspended=<CLEAR>
- item={0}<ENTER>
- itemQty={0}<QTY>
- itemPrice={0}<PRICE>
- itemDealPrice={0}<SLASH>{1}<PRICE>
- itemScanned=<SLASH>{0}<ENTER>
- itemVoid=<VOID>{0}<ENTER>
- itemVoidScanned=<VOID><SLASH>{0}<ENTER>
- itemReturn=<REFUND>{0}<ENTER>
- itemReturnScanned=<REFUND><SLASH>{0}<ENTER>
- itemWeight={0}<WEIGHT>
- logoff1=<SIGNON>
- logoff2=<SIGNON>
- managerOverride=<OVERRIDE>{0}<ENTER>
- operatorOverride=<OVERRIDE><ENTER>
- password={0}<SIGNON>
- retrieveTransaction=<SUSPENDRET>{0}<SLASH>{1}<ENTER>
- specialSignonToOverride=<SIGNON>
- startTransaction=0<SIGNON>

- suspendTransaction=<TOTAL><SUSPENDRET><TOTAL>
- tenderVariety={0}<SLASH>
- total=<TOTAL>
- trainingModeOn=7<SIGNON>
- transDataEntry=<DATAENTRY>{0}<ENTER>
- void=<VOID>
- verifySignature=<ENTER>
- voidTransaction=<VOID><TOTAL>

nextFile=userseq - This property is used to tell AEF the name of the next property file in the bundle chain.

Logon Bundle

The Logon bundle is used to specify a default user ID and password combination for specific terminals. The files that makeup the Logon bundle are applogon.properties, and userlogon.properties.

For each property, a key is provided followed by an equal sign, a number, a comma, and another number. The key is the terminal number. The first number after the equal sign is the operator ID to use. The number after the comma is the password to use.

Applogon.properties (ACE)

The applogon.properties file includes the following properties.

- 001=1,1
- 002=1,1
- 100=1,1
- 999=2,2

nextFile=userlogon - This property is used to tell AEF the name of the next property file in the bundle chain.

Applogon.properties (GSA)

The applogon.properties file includes the following properties.

- 001=1,1
- 002=1,1
- 100=1,1
- 999=2,2

nextFile=userlogon - This property is used to tell AEF the name of the next property file in the bundle chain.

Applogon.properties (SA)

The applogon.properties file includes the following properties.

- 001=1,1
- 002=1,1
- 100=1,1
- 999=2,2

nextFile=userlogon - This property is used to tell AEF the name of the next property file in the bundle chain.

Session Bundle

The Session bundle is used to configure session specific settings. The files that makeup the Session bundle are session.properties, appsession.properties, and usersession.properties.

For each property, a key is provided followed by an equal sign and a value.

Session.properties

The session.properties file includes the following properties.

- **aefio.device.group=default** – The AEFIO Device Group defaults to com.ibm.retail.AEF.io.emujpos for emulated devices and jpos.res.jpos for real devices.
- **session.extensions=** - The AEF will instantiate an instance of each class listed (comma separated). The classes should implement the com.ibm.retail.AEF.session.SessionExtension interface. The "setSession" method will be called for each session after the session has become ready. The AEFSession object will be passed to the extension on the "setSession" method. Example: session.extensions=com.xyz.foo.SessionFoo An instance of SessionFoo will be instantiated with each AEFSession. NOTES: 1. Extension class must use a default constructor for instantiation. 2. Extension class must implement the com.ibm.retail.AEF.session.SessionExtension interface. 3. Session extension objects are created after "SessionReady" event in AEFSession. 4. The AEFSession object will be passed to the extension via the "setSession" method.
- **trace.buffer.size=1000** - SessionTrace properties.
- **trace.file.count=10** - SessionTrace properties.
- **trace.file.pattern=R::C:/silogs/%n.%g** - SessionTrace properties.
- **trace.level=COARSE** - SessionTrace properties. Valid trace levels: FINE, MEDIUM, COARSE or DUMP
- **trace.autodump=OFF** - SessionTrace properties.
- **trace.default.encoding=UTF-8** - SessionTrace properties.
- **display.debug.gui=false** - Indicates whether a 2x20 GUI should be displayed for the session.
- **start.pos.sales.application=true** - Tells the session to start the POS application
- **css.backup.nvram.enabled=false** - Setting this to true will save hard totals to disk. Persistent hard totals are required for transaction recovery. (CSS)
- **css.backup.nvram.offsets=** - Setting css.backup.nvram.offsets will restrict the backup of hard totals to include only the offsets specified, providing a more limited backup of totals. This is useful to provide limited transaction recovery (e.g, payment processing) while reducing the performance impact of writing all totals to disk.

nextFile=appsession - This property is used to tell AEF the name of the next property file in the bundle chain.

Appsession.properties (ACE)

The appsession.properties file includes the following properties.

- **pos.sales.application=R::h0:/adx_ipgm/JSIFTS10.386** - Terminal Session application settings (CSS only)
- **css.nvram.prime.file=c:/adx_idt1/nvrprime.dat** The contents of the NVRAM emulated memory are primed from this file.

nextFile=usersession - This property is used to tell AEF the name of the next property file in the bundle chain.

Appsession.properties (GSA)

The appsession.properties file includes the following properties.

- **pos.sales.application=R::h0:/adx_ipgm/EALTS10L.286** - Terminal Session application settings (CSS only)

nextFile=usersession - This property is used to tell AEF the name of the next property file in the bundle chain.

Appsession.properties (SA)

The appsession.properties file includes the following properties.

- **pos.sales.application=R::h0:/adx_ipgm/EAMTS10L.286** - Terminal Session application settings (CSS only)

nextFile=usersession - This property is used to tell AEF the name of the next property file in the bundle chain.

State Bundle

The State bundle is used to map a logical application state to a state number. The files that makeup the State bundle are states.properties, appstate.properties, and userstates.properties.

For each property, a key is provided followed by an equal sign and a value. The values for these keys must match the valid state values for the base application.

States.properties

The state.properties file includes the following properties.

nextFile=appstate – This property is used to tell AEF the name of the next property file in the bundle chain.

Appstate.properties (ACE)

The appstate.properties file includes the following properties.

- CLEAR=1
- ID=2
- PASSWORD=3
- SO_PASSWORD=4
- SECURED=5

- OVERRIDE=6
- ACCT_NUMBER=7
- ACCT_OVERRIDE=8
- ENTER_CLEAR=9
- ITEMENTRY=10
- MAIN=10
- ENTER_DATA=11
- STANDALONE=12
- ENTER=13
- EFT_ACCOUNT=15
- EFT_BLINQ=16
- TENDER_CASHING_EXCHANGE=20
- CASHIER_LOAN_PICKUP_TENDER_COUNT=21
- TENDER_LIST=22
- TERMINAL_TRANSFER_MONITOR=23
- PRICE_VERIFY_CHANGE=24
- ALLKEYS=25
- EFTVOID2=26
- MICRTRAN=27
- MICRACCT=28
- MICRSEQN=29
- AUTOSIGN=55

nextFile=userstate – This property is used to tell AEF the name of the next property file in the bundle chain.

Appstate.properties (GSA)

The appstate.properties file includes the following properties.

- ID=1
- PASSWORD=2
- TRANSACTION_SELECT=3
- ACCT_NUMBER=4
- TERMS_OF_SALE=5
- CREDIT_AUTHORIZATION=6
- ITEMENTRY=7
- ENTER_CLASS_STOCK=8
- ENTER_STOCK=9
- ENTER_QTY_AMOUNT=10
- ENTER_AMOUNT=11
- ENTER_PAYMENT_AMOUNT=12
- ENTER_ALLOWANCE_AMOUNT=13
- ENTER_TAXCODE_AMOUNT=14
- ENTER_DISCOUNT_RATE=15
- ENTER_DEPARTMENT_AMOUNT=16
- ENTER_TENDER_AMOUNT=17
- ENTER_ACCT_AMOUNT=18
- ENTER_TENDER_AMOUNT_3=19
- ORIGINAL SALESPERSON=20
- ENTER_FEE_AMOUNT=21
- STOCK_REQUIRED=22
- DECIMAL_QUANTITY_AMOUNT=23
- ENTER_DEPOSIT_AMOUNT=24
- ENTER_LOAN_AMOUNT=25

- ENTER_WITHDRAWAL=26
- ENTER_CASH_COUNT=27
- TOTALS_READOUT_RESET=28
- ENTER_DEPARTMENT_FOR_TOTALS=29
- ENTER_ITEM_FOR_MOVEMENT=30
- PRICE_CHANGE=31
- SET_TRANSACTION=32
- REENTRY_OFFLINE_TERM=33
- REENTRY_OFFLINE_DATE=34
- REENTRY_OFFLINE_TRANS_NUMBER=35
- DECIMAL_QUANTITY_ITEM=36
- ENTER_LAYAWAY_ACCT=37
- LAYAWAY_CANCEL=38
- VERIFY_PAYMENT=39
- CLEAR=40
- SIGNOFF=41
- TOTALS_READOUT_RESET_CODE=42
- TENDER_LISTING_SINGLE_ALL=43
- TENDER_LISTING_OPERATOR_TERMINAL=44
- TENDER_LISTING_HALT_PRINT=45
- TENDER_TYPE_TO_REMOVE=46
- DECIMAL_QUANTITY_TENDERED=47
- RETRIEVE=48
- QUANTITY_REQUIRED=49
- PRICE_REQUIRED=50
- ACCT_REQUIRED=51
- DISCOUNT=52
- LAYAWAY_PAYMENT=53
- CASH_COUNT_2=54
- AMOUNT_TENDER_4=55
- ENTER_ACCT_A=56
- ENTER_ACCT_B=57
- ENTER_ACCT_C=58
- ENTER_ACCT_D=59
- MODIFY_TICKET_QTY_AMT=60
- MODIFY_TICKET_AMOUNT=61
- MODIFY_TICKET_WAND_SCAN=62
- REENTER_VOID=63
- TOTAL_LIST=75
- KEYBATCH_NUMBER=76
- TERMINAL_MONITOR=77
- ENTER_ONLY=78
- TERMINAL_SIGNOFF=79
- EFT_RETRY=80
- EFT_ACCT=81
- EFT_EXPIRATION_DATE=82
- EFT_DEBIT_AMOUNT=83
- EFT_PROCEED_CANCEL=84
- VERIFY_CANCEL=85
- LCOFFLTS=86
- LCOFFLBD=87
- LCOFFLFC=88
- LCOFFLDM=89
- LPAYOFFL=90
- CWCLR=91
- MSREENTRY=92

- EFTHOST=93
- ENT-CLR=94
- ACCT-OVR=95
- ACCTOPT=96
- CPVERSIG=97
- AGERESTRICTED=99
- EXCHANGE=130
- EXCHANGE_AMOUNT=131
- EXCHCNVT=132
- TDRLIST2=133
- TDRLIST3=134
- TDRLOAN=135
- CVTBALDU=137
- MICR=138
- MCF01=140
- DUPREC=149

nextFile=userstate – This property is used to tell AEF the name of the next property file in the bundle chain.

Appstate.properties (SA)

The appstate.properties file includes the following properties.

- CLEAR=1
- ID=2
- PASSWORD=3
- SO_PASSWORD=4
- SECURED=5
- OVERRIDE=6
- ACCT_NUMBER=7
- ACCT_OVERRIDE=8
- ENTER_CLEAR=9
- ITEMENTRY=10
- MAIN=10
- ENTER_DATA=11
- STANDALONE=12
- ENTER=13
- EFT_ACCOUNT=15
- EFT_BLINQ=16
- TENDER_CASHING_EXCHANGE=20
- CASHIER_LOAN_PICKUP_TENDER_COUNT=21
- TENDER_LIST=22
- TERMINAL_TRANSFER_MONITOR=23
- PRICE_VERIFY_CHANGE=24
- EFTVOID1=25
- EFTVOID2=26
- MICRTRAN=27
- MICRACCT=28
- MICRSEQN=29
- AUTOSIGN=55

nextFile=userstate – This property is used to tell AEF the name of the next property file in the bundle chain.

Substate Bundle

The Substate bundle is used to map a logical application substate to a substate number. The files that makeup the Substate bundle are appsubstates.properties, and usersubstates.properties.

For each property, a key is provided followed by an equal sign and a value. These values must match the valid substate values for the base application.

Appsubstates.properties (ACE)

For ACE, additional keys have been added to match a substate to an error message. The appsubstates.properties file includes the following properties.

- EXPECTING_ITEM=1001
- EXPECTING_TENDER=1002
- EXPECTING_FOODSTAMP_TENDER=1003
- EXPECTING_FOREIGN_TENDER=1004
- EXPECTING_COUPON=1005
- EXPECTING_MANAGER_OVERRIDE=1006
- EXPECTING_OPERATOR_OVERRIDE=1007
- EXPECTING_PROCEDURE=1008
- 10001=B001 - B001 NO COUPON ON
- INVALID_KEY_SEQUENCE=10003 - B003 CHECK KEY SEQUENCE
- 10003=B003 - B003 CHECK KEY SEQUENCE
- 10004=B004 - B004 MANAGER'S KEY IS REQUIRED
- 10005=B005 - B005 REMOVE MANAGER'S KEY
- KEYED_DATA_OUT_OF_RANGE=10006 - B006 KEYED DATA OUT OF RANGE
- 10006=B006 - B006 KEYED DATA OUT OF RANGE
- 10007=B007 - B007 CHECK THE PRINTER
- 10008=B008 - B008 %1 TRANSACTION LIMIT CHECK
- 10009=B009 - B009 ITEM LIMIT CHECK
- 10010=B010 - B010 USE LOOKUP KEYS
- 10011=B011 - B011 CHECK COIN DISPENSER
- 10012=B012 - B012 CHECK JOURNAL PAPER
- CLOSE_CASH_DRAWER=10013 - B013 CLOSE THE CASH DRAWER
- 10013=B013 - B013 CLOSE THE CASH DRAWER
- 10014=B014 - B014 NO TRANSACTION RECOVERY
- 10015=B015 - B015 CHECK THE KEYBOARD
- 10016=B016 - B016 WRITE ERROR TRANSACTION LOST
- 10017=B017 - B017 EXTENDED PRICE LIMIT CHECK
- 10018=B018 - B018 NO WEIGHT ON
- 10019=B019 - B019 WEIGHT NEEDED
- 10020=B020 - B020 TAKE TOTAL
- 10021=B021 - B021 CHECK TARE CODE
- 10022=B022 - B022 NO DEPOSIT ON
- 10023=B023 - B023 NO QUANTITY ON
- 10024=B024 - B024 PRICE NEEDED
- 10025=B025 - B025 SELL ITEM WITHOUT REPEAT KEY
- 10026=B026 - B026 ITEM NOT FOUND
- 10027=B027 - B027 NOT FOR SALE
- 10028=B028 - B028 TAKE FOODSTAMP TOTAL
- 10029=B029 - B029 QUANTITY NEEDED
- 10030=B030 - B030 TRANSACTION TOTAL IS TOO LARGE
- 10031=B031 - B031 NO F.S. KEY ON
- 10032=B032 - B032 NO TAX KEY ON
- 10033=B033 - B033 SCALE ERROR CLEAR/RETRY

- 10034=B034 - B034 CHECK TENDER VARIETY
- 10035=B035 - B035 CHECK TENDER DENOMINATION
- 10036=B036 - B036 CHECK QUANTITY
- 10037=B037 - B037 CHECK PRICE
- 10038=B038 - B038 CHECK WEIGHT
- 10039=B039 - B039 CHECK ITEM DATA
- 10040=B040 - B040 FILE ACCESS FAILED
- 10041=B041 - B041 CHECK ACCOUNT DATA
- 10042=B042 - B042 ITEM IS ON THE SCALE
- 10043=B043 - B043 NO REFUND ON
- 10045=B045 - B045 %1 DISCOUNT LIMIT CHECK
- 10046=B046 - B046 ONE DISCOUNT PER TRANSACTION
- 10047=B047 - B047 CHECK DISCOUNT GROUP NUMBER
- ACCOUNT_NUMBER_NEEDED=10048 - B048 ACCOUNT NUMBER NEEDED
- 10048=B048 - B048 ACCOUNT NUMBER NEEDED
- INSERT_DOCUMENT_TO_BE_FRANKED=10049 - B049 INSERT DOCUMENT TO BE FRANKED
- 10049=B049 - B049 INSERT DOCUMENT TO BE FRANKED
- 10050=B050 - B050 TENDER IS VERIFIED
- 10051=B051 - B051 NOT ON FILE
- 10052=B052 - B052 ACCOUNT LIMIT EXCEEDED (Quantity)
- 10053=B053 - B053 ACCOUNT LIMIT EXCEEDED (Amount)
- 10054=B054 - B054 ACCOUNT LIMIT EXCEEDED (Total Amount)
- 10055=B055 - B055 RISK 1
- 10056=B056 - B056 RISK 2
- 10057=B057 - B057 RISK 3
- 10058=B058 - B058 RISK 4
- CHECK_OVERRIDE_NUMBER=10059 - B059 CHECK OVERRIDE NUMBER
- 10059=B059 - B059 CHECK OVERRIDE NUMBER
- 10060=B060 - B060 CHANGE AMOUNT LIMIT CHECK
- 10061=B061 - B061 PRICE TOO LARGE
- 10062=B062 - B062 KEYED PRICE LIMIT CHECK
- 10063=B063 - B063 TENDER AMOUNT LIMIT CHECK
- 10064=B064 - B064 COUPON MUST MATCH PREVIOUS SALE
- 10065=B065 - B065 COUPON VALUE EXCEEDS ITEM VALUE
- 10066=B066 - B066 RETURN COUPON BEFORE ITEM CANCEL
- 10067=B067 - B067 CANCEL MUST MATCH PREVIOUS ENTRY
- 10068=B068 - B068 FULL KEYING SEQUENCE REQUIRED
- 10069=B069 - B069 TENDERS/COUPONS MUST BE RETURNED
- 10070=B070 - B070 NEW PASSWORD ACCEPTED
- 10071=B071 - B071 PASSWORD MUST MATCH PREVIOUS ENTRY
- 10072=B072 - B072 PASSWORD NEEDED
- 10073=B073 - B073 NOT AVAILABLE OFFLINE
- 10074=B074 - B074 TERMINAL HAS BEEN TRANSFERRED
- 10075=B075 - B075 CHECK PROCEDURE NUMBER
- 10076=B076 - B076 AUTHORIZATION REQUIRED
- 10077=B077 - B077 CHECK PASSWORD
- 10078=B078 - B078 CHECK OPERATOR NUMBER
- 10079=B079 - B079 OPERATOR STILL ACTIVE ON %1
- 10080=B080 - B080 KEY OPERATOR ,*,PASSWORD,SIGN-ON
- 10081=B081 - B081 CLOSING PERIOD WAIT AND RETRY
- 10082=B082 - B082 PICKUP NEEDED SOON
- 10083=B083 - B083 PICKUP REQUIRED NOW
- 10084=B084 - B084 NOT AVAILABLE WHILE TRAINING
- 10085=B085 - B085 TILL EXCHANGE REQUIRED
- 10086=B086 - B086 MONITOR ALREADY ACTIVE ON %1

- 10087=B087 - B087 TERMINAL MUST MATCH PREVIOUS ENTRY
- 10088=B088 - B088 CASH CHANGE OWED TO CUSTOMER
- 10089=B089 - B089 CASH CHANGE NEEDED FROM CUSTOMER
- 10090=B090 - B090 CHECK TENDER TYPE
- 10092=B092 - B092 INVALID FEE AMOUNT
- 10094=B094 - B094 PROCEDURE NOT AVAILABLE NOW
- 10096=B096 - B096 CHECK TERMINAL NUMBER
- 10098=B098 - B098 CHECK PIN PAD
- 10099=B099 - B099 INVALID REASON CODE %1
- 10100=B100 - B100 RECURSIVE LINK CHAIN
- 10101=B101 - B101 COUPON AS FIRST ITEM NOT ALLOWED
- 10103=B103 - B103 ERROR WRITING TRANSACTION
- 10104=B104 - B104 EAS ITEM
- 10106=B106 - B106 KEYED ENTRY NOT ALLOWED
- 10110=B110 - B110 ONLY %1 DIGITS ALLOWED
- 10111=B111 - B111 INVALID PRINT REQUESTED
- 10112=B112 - B112 REMOVE CHECK FROM PRINTER
- 10113=B113 - B113 REMOVE FRANKED TENDER
- 10117=B117 - B117 PLEASE END TRANSACTION
- 10118=B118 - B118 NO MORE ITEMS ALLOWED
- 10132=B132 - B132 PREVIOUS ITEM STILL ON SCALE
- 10133=B133 - B133 PLEASE RE-WEIGH ITEM
- 10140=B140 - B140 NO RECEIPT TO REPRINT
- 10141=B141 - B141 INVALID DATE
- INVALID_DATE=10141 - B141 INVALID DATE
- 10142=B142 - B142 NON-WIC ITEM
- 10143=B143 - B143 CHECK FORMAT NOT UNDERSTOOD
- 10144=B144 - B144 THIS IS NOT A WIC TENDER
- 10145=B145 - B145 WIC TENDER NOT ALLOWED
- 10146=B146 - B146 TARE WEIGHT TOO LARGE
- 10147=B147 - B147 INVALID ID TYPE
- 10148=B148 - B148 CHECK CONFIGURATION
- 10149=B149 - B149 INVALID ENTRY
- 10150=B150 - B150 VOUCHER EXPIRED
- 10151=B151 - B151 TENDER NOT LOANED
- 10152=B152 - B152 TENDER NOT PICKED UP
- 10169=B169 - B169 TAKE FOREIGN TOTAL
- 10170=B170 - B170 MISSING TENDER DETAILS
- 10172=B172 - B172 NEGATIVE BALANCE
- 10173=B173 - B173 ITEM ENTERED TAKE TOTAL
- 10174=B174 - B174 COUPON SCANNED TAKE TOTAL
- 10175=B175 - B175 KEYED ACCOUNT REQUIRES OVERRIDE
- 10176=B176 - B176 TAX EXEMPT DISCGROUP NOT ALLOWED
- 10177=B177 - B177 NOT DSCNTABLE
- 10178=B178 - B178 ITEM DISCOUNTS NOT ALLOWED
- 10179=B179 - B179 VERIFY ENTERED PRICE
- 10180=B180 - B180 VOUCHER NOT YET VALID
- 10181=B181 - B181 PLEASE CHECK VOUCHER DATES
- 10182=B182 - B182 DATE MUST BE 8 DIGITS
- 10183=B183 - B183 VOID EBT FOODSTAMPS
- 10184=B184 - B184 NO FUEL ON
- 10185=B185 - B185 FUEL NEEDED
- 10186=B186 - B186 CHECK FUEL
- 10187=B187 - B187 CHECK EXTENDED PRICE
- 10188=B188 - B188 NO XPRICE ON
- 10300=B300 - B300 TOO MANY EPS TENDERS

- 10301=B301 - B301 USE ANOTHER TENDER
- 10302=B302 - B302 PIN PAD NOT AVAILABLE
- 10303=B303 - B303 CHECK TENDER TYPE
- 10304=B304 - B304 CUSTOMER MUST SIGN DOCUMENT
- 10305=B305 - B305 INVALID DATE
- 10306=B306 - B306 INVALID ACCOUNT NUMBER
- 10307=B307 - B307 BALANCE INQUIRY FOR EBT ONLY
- 10308=B308 - B308 CUSTOMER CHANGED AMOUNT
- 10309=B309 - B309 APPROVED
- 10310=B310 - B310 REFUND NOT ALLOWED
- 10311=B311 - B311 NOT AUTHORIZED
- 10312=B312 - B312 TOO LONG IN OFFLINE MODE
- 10313=B313 - B313 OFFLINE COUNT EXCEEDED
- 10314=B314 - B314 OFFLINE AMOUNT EXCEEDED
- 10315=B315 - B315 OFFLINE NOT ALLOWED
- 10316=B316 - B316 RISK CODE 1
- 10317=B317 - B317 FLOOR LIMIT 2 EXCEEDED
- 10318=B318 - B318 CARD HAS EXPIRED
- CARD_HAS_EXPIRED=10318 - B318 CARD HAS EXPIRED
- 10319=B319 - B319 CHECK EPS CONFIGURATION
- 10320=B320 - B320 INVALID DATA RETRY CARD SWIPE
- 10321=B321 - B321 CARD TYPE NOT KNOWN
- 10322=B322 - B322 FLOOR LIMIT 1 EXCEEDED
- 10323=B323 - B323 CALL FOR AUTHORIZATION
- 10324=B324 - B324 CANNOT COMPLETE
- 10325=B325 - B325 CUSTOMER CHOSE %1
- 10326=B326 - B326 CUSTOMER CANCELLED TENDER
- 10327=B327 - B327 OUT OF MEMORY GIVE CASH FOR TENDER
- 10328=B328 - B328 VOID FAILED - GIVE CASH FOR TENDER
- 10329=B329 - B329 SECOND ID NEEDED
- 10330=B330 - B330 KEY ENTRY NOT ALLOWED
- 10331=B331 - B331 MSR NOT ALLOWED
- 10332=B332 - B332 MICR NOT ALLOWED
- 10333=B333 - B333 NO EBT TENDERS CONFIGURED
- 10334=B334 - B334 VOUCHER REQUIRED
- 10335=B335 - B335 UNABLE TO READ CARD
- 10336=B336 - B336 NUMBER IS NOT VALID
- 10337=B337 - B337 SCANNED ENTRY NOT ALLOWED
- 10338=B338 - B338 INTERNAL EPS ERROR
- 10339=B339 - B339 ENTER TO ACCEPT CLEAR TO REJECT
- 10340=B340 - B340 PAYMENT TYPE MISMATCH
- 10341=B341 - B341 WORKING KEY REQUIRED
- 10342=B342 - B342 OPTIONS ERROR ID SPECIFIED
- 10343=B343 - B343 OPTIONS ERROR INVALID ID DEVICE
- 10344=B344 - B344 OPTIONS ERROR SECOND ID UNDEFINED
- 10345=B345 - B345 OPTIONS ERROR ID TYPE NOT FOUND
- 10346=B346 - B346 ERROR LOADING PIN PAD DEVICE
- 10347=B347 - B347 PIN PAD NOT CAPABLE
- 10348=B348 - B348 PIN PAD PARAMETERS ERROR
- 10349=B349 - B349 ZERO AMOUNT IS NOT ALLOWED
- 10350=B350 - B350 PIN PAD NOT ALLOWED
- 10351=B351 - B351 GET CUSTOMER OK
- 10352=B352 - B352 GIFT CARD AMT 0 COLLECT TO RECYCLE
- 10353=B353 - B353 DUPLICATE VALUE CARD FOUND IN ORDER
- 10354=B354 - B354 VALUE CARD ITEM REMOVED FROM ORDER
- 10355=B355 - B355 ADDED TO ORDER %1

- 10356=B356 - B356 NO MATCHED ITEM VOID NOT ALLOWED
- 10357=B357 - B357 PREVIOUS HOST REQUEST PENDING
- 10358=B358 - B358 VOID FROM ORDER %1
- 10359=B359 - B359 USE ANOTHER TENDER (GIPC DOWN)
- 10360=B360 - B360 VALUE CARDS MUST BE VOIDED
- 10361=B361 - B361 CARD REFUNDED %1
- 10362=B362 - B362 VALUE CARD ITEM VOID NOT APPROVED
- 10363=B363 - B363 VALUE CARD ITEM REFUND NOT APPROVED
- 10364=B364 - B364 CARD RETURNED %1
- 10365=B365 - B365 VALUE CARD ITEM RETURN NOT APPROVED
- 10366=B366 - B366 CASHBACK GIVEN ONLY ON FINAL TENDER
- 10367=B367 - B367 USE VOID OF REF FOR THE VALUE CARD
- 10368=B368 - B368 USE VOID FOR THIS VALUE CARD
- 10369=B369 - B369 SIGNATURE NOT AVAILABLE
- 10370=B370 - B370 OVERRIDE NEEDED GIPC DOWN
- 10371=B371 - B371 CANNOT PROCESS VALUE CARD-GIPC DOWN
- 10372=B372 - B372 BALANCE INQUIRY NOT ENABLED
- 10373=B373 - B373 GIFT CARD NOT CONFIGURED
- 10383=B383 - B383 DATA LENGTH ERROR
- INVALID_CARD_VALIDATION_NUMBER=10383 - B383 DATA LENGTH ERROR
- 10600=B600 - B600 POINTS ENTRY REQUIRES CUSTOMER #
- 10601=B601 - B601 HOME STORE MUST REDEEM POINTS
- 10602=B602 - B602 MORE POINTS NEEDED FOR DISCOUNT
- 10603=B603 - B603 NO AVAILABLE DRAWER
- 10604=B604 - B604 CHECK KEYED CUSTOMER NUMBER
- 10605=B605 - B605 CHECK CUSTOMER NUMBER %1
- 10606=B606 - B606 REDEMPTION MUST BE POSITIVE
- 10607=B607 - B607 BONUS POINT LIMIT CHECK
- 10608=B608 - B608 ASK FOR CARD CLEAR TO CONTINUE
- 10609=B609 - B609 CHECK KEYED COUPON NUMBER
- 10610=B610 - B610 LIMITED NUMBER OF COUPONS PER ORDER
- 10611=B611 - B611 COUPON HAS EXPIRED
- 10612=B612 - B612 MINIMUM SALE NOT SATISFIED
- 10613=B613 - B613 PLEASE WRITE %1 ON COUPON
- 10614=B614 - B614 ERROR ACCESSING CUSTOMER FILE
- 10615=B615 - B615 COUPON REQUIRES CUSTOMER NUMBER
- 10616=B616 - B616 PREFERRED PROCESSING INACTIVE
- 10617=B617 - B617 COUPON DOES NOT MATCH
- 10618=B618 - B618 DAILY CARD USES EXCEEDED LIMIT
- 10619=B619 - B619 CUSTOMER NOT AT REQUIRED LEVEL
- 10671=B671 - B671 FUNCTION IS NOT AVAILABLE
- 10672=B672 - B672 RETRY WITH RETRIEVAL NUMBER
- 10673=B673 - B673 NO. OF SUSPENDS EXCEEDED
- 10676=B676 - B676 ACTIVE SUSPENDS CLEAR OR CONTINUE
- 10677=B677 - B677 RETRY WHERE SUSPENDED
- 10678=B678 - B678 UNABLE TO RETRIEVE DATA
- 10679=B679 - B679 UNABLE TO UPDATE RETRIEVED RECORD
- 10680=B680 - B680 TRANSACTION RETRIEVED ALREADY
- 10683=B683 - B683 SAME OPERATOR MUST RETRIEVE
- 10684=B684 - B684 FORCE RETRIEVE REQUIRED
- 10685=B685 - B685 SUSPENSION INCOMPLETE
- 10686=B686 - B686 RETRIEVAL INCOMPLETE
- 10687=B687 - B687 TRAINING MODE MISMATCH
- 10688=B688 - B688 OVERRIDE NEEDED FOR SUSPEND
- 10689=B689 - B689 OVERRIDE NEEDED FOR RETRIEVE
- 10690=B690 - B690 %1 RESTRICTED ITEMS

- 10691=B691 - B691 VOID %1 ITEMS
- 10864=B864 - B864 CHECK FORMAT NOT UNDERSTOOD
- 10865=B865 - B865 ERROR READING CHECK
- 10870=B870 - B870 VERIFY THE SIGNATURE ON CHECK
- 10872=B872 - B872 PRINTER COVER OPEN
- 10873=B873 - B873 INSERT CHECK TO BE PRINTED
- 10874=B874 - B874 REMOVE CHECK FROM PRINTER
- 10876=B876 - B876 PRESS DOCUMENT INSERT BUTTON
- 10877=B877 - B877 CHANGE PASSWORD
- 10878=B878 - B878 PASSWORD EXPIRES IN %1 DAYS
- 10879=B879 - B879 PASSWORD NOT CHANGED
- 10880=B880 - B880 PRICE CHANGE NOT ALLOWED
- 10893=B893 - B893 PRINTER OUT OF PAPER/PRINT COVER UP
- 10894=B894 - B894 INVALID PRINTER COMMAND
- 10901=B901 - B901 DEVICE OFFLINE
- 10902=B902 - B902 DISPLAY OFFLINE
- 10903=B903 - B903 VIDEO DISPLAY OFFLINE
- 10904=B904 - B904 SCALE OFFLINE
- 10905=B905 - B905 KEYBOARD OFFLINE
- 10906=B906 - B906 TONE OFFLINE
- 10907=B907 - B907 MSR OFFLINE
- 10908=B908 - B908 CASH DRAWER OFFLINE
- 10909=B909 - B909 UNKNOWN DEVICE OFFLINE
- 10910=B910 - B910 UNABLE TO OPEN COIN DISPENSER
- 10911=B911 - B911 RESTRICTED QTY %2
- 10999=B999 - B999 PRESS CLEAR VALID TAX CODES ARE 1-4
- EXPECTING_PRICE=11033 - 1033 ENTER PRICE OR PRESS CLEAR
- 11033=11033 - 1033 ENTER PRICE OR PRESS CLEAR
- EXPECTING_QUANTITY=11034 - 1034 ENTER QUANTITY OR PRESS CLEAR
- 11034=11034 - 1034 ENTER QUANTITY OR PRESS CLEAR
- EXPECTING_WEIGHT=11035 - 1035 ENTER WEIGHT OR PRESS CLEAR
- 11035=11035 - 1035 ENTER WEIGHT OR PRESS CLEAR
- REPEAT_SIGNOFF=11036 - 1036 REPEAT SIGN-OFF KEY OR PRESS CLEAR
- 11036=11036 - 1036 REPEAT SIGN-OFF KEY OR PRESS CLEAR
- EXPECTING_REPEAT_TENDER=11037 - 1037 REPEAT TENDER KEY OR PRESS CLEAR
- SELECT_PROCEDURE=11038 - 1038 SELECT PROCEDURE OR ENTER NEW ITEM
- EXPECTING_PASSWORD_REENTRY=11039 - 1039 VERIFY NEW PASSWORD BY RE-ENTERING IT
- ENTER_ITEM=11041 - 1041 ENTER ITEM
- CLOSED=11042 - 1042 ***** CLOSED *****
- TERMINAL_SECURED=11043 - 1043 **TERMINAL SECURED**
- TRAINING_MODE=11045 - 1045 ** TRAINING **
- EXPECTING_WIC_ITEM_ENTRY=11046 - 1046 ENTER ITEM ** WIC **
- TRANSACTION_VOIDED=11048 - 1048 *VOID TRANSACTION*
- EXPECTING_VERIFICATION_RETRY=11055 - 1055 ENTER TO RETRY CLEAR TO BYPASS
- EXPECTING_RESPONSE_FROM_HOST=11056 - 1056 PROCESSING REQUEST PLEASE WAIT
- EXPECTING_ACCOUNT_NUMBER=11057 - 1057 ENTER ACCOUNT NUMBER OR CLEAR
- 11057=11057 - 1057 ENTER ACCOUNT NUMBER OR CLEAR
- EXPECTING_PINPAD_INPUT=11058 - 1058 WAITING FOR CUSTOMER INPUT
- EXPECTING_OK_OR_CANCEL=11060 - 1060 ENTER TO CONTINUE TOTAL TO CANCEL

- EXPECTING_DATE_OF_BIRTH=11076 - 1076 ENTER DATE OF BIRTH MMDDYYYY
- 11076=11076 - 1076 ENTER DATE OF BIRTH MMDDYYYY
- EXPECTING_VOID_REASON_CODE=11516 - 1516 ENTER VOID REASON CODE OR PRESS CLEAR
- 11516=11516 - 1516 ENTER VOID REASON CODE OR PRESS CLEAR
- EXPECTING_EXPIRATION_DATE=11162 - 1162 ENTER EXP DATE MMY
- 11162=11162 - 1162 ENTER EXP DATE MMY
- EXPECTING_TENDER_CASHING=11195 - 1195 KEY TENDER SEQUENCE TO CASH TENDER
- EXPECTING_TENDER_CORRECTION=11196 - 1196 KEY TENDER CORRECTION
- EXPECTING_LOAN_AMOUNT=11197 - 1197 KEY LOAN AMOUNTS WITH TENDER KEYS
- EXPECTING_PICKUP_AMOUNT=11198 - 1198 KEY PICKUP AMOUNTS WITH TENDER KEYS
- ITEM_TRAINING=11201 - 1201 ENTER ITEM ** TRAINING **
- EXPECTING_TERMINAL_TO_TRANSFER=11202 - 1202 ENTER ADDRESS OF TERMINAL TO TRANSFER
- EXPECTING_TERMINAL_TO_MONITOR=11203 - 1203 ENTER ADDRESS OF TERMINAL TO MONITOR
- EXPECTING_SECOND_EXCHANGE_TENDER=11206 - 1206 KEY NEW TENDER OR PRESS TOTAL
- EXPECTING_VOID_TENDER_FEE=11207 - 1207 ENTER VOIDED FEE OR PRESS CLEAR
- TENDER_LISTING=11208 - 1208 TENDER FOR SINGLE/TOTAL FOR ALL LISTS
- EXPECTING_VOUCHER_ISSUE_DATE=11210 - 1210 VOUCHER ISSUE DATE %06.6s
- EXPECTING_VERIFY_ADDRESS=11211 - 1211 VERIFY ADDRESS BY RE-ENTERING IT
- EXPECTING_TRANSFER_AS_SAME_OPER=11212 - 1212 TOTAL FOR THIS OPER ENTER FOR SAME OPER
- RECOVERED_TRANSACTION=11213 - 1213 RECOVERING ORDER VOID OR CONTINUE
- EXPECTING_CUSTOMER_ID=11217 - 1217 ENTER CUSTOMER ID
- EXPECTING_VOUCHER_EXPIRATION=11223 - 1223 VOUCHER EXPIRATION DATE %06.6s
- TENDER_LIST_SELECTION=11224 - 1224 ENTER FOR SINGLE OR TOTAL FOR ALL LISTS
- TENDER_FEE=11225 - 1225 ENTER TENDER FEE OR PRESS CLEAR
- EXPECTING_MONITOR=11227 - 1227 WAITING FOR CASHIER TO START NEW ORDER
- EXPECTING_STATE_ID=11228 - 1228 ENTER STATE ID
- EXPECTING_CHECK_NUMBER=11229 - 1229 ENTER CHECK NUMBER
- EXPECTING_TRANSIT_CODE=11261 - 1261 ENTER TRANSIT CODE
- EXPECTING_DRIVERS_LICENSE_NUMBER=11264 - 1264 ENTER D/L NUMBER
- EXPECTING_CHECK_ACCOUNT=11266 - 1266 ENTER CHECK ACCOUNT
- EXPECTING_FUEL=11507 - 1507 ENTER FUEL VOLUME OR PRESS CLEAR
- EXPECTING_PRICE_RC=11514 - 1514 ENTER PRICE REASON CODE OR PRESS CLEAR
- REMOVE_CUSTOMER_RECEIPT=13008 - 3008 REMOVE CUSTOMER RECEIPT
- 13008=13008 - 3008 REMOVE CUSTOMER RECEIPT
- TENDER_APPROVED=13010 - 3010 APPROVED
- EXPECTING_AUTHORIZATION_CODE=13013 - 3013 ENTER AUTHORIZATION CODE
- 13013=13013 - 3013 ENTER AUTHORIZATION CODE

- EXPECTING_BALANCE_INQUIRY=13020 - 3020 BALANCE INQUIRY 1=EBT 2=CASH
- EXPECTING_VOUCHER_NUMBER=13031 - 3031 ENTER VOUCHER NUMBER
- 13031=13031 - 3031 ENTER VOUCHER NUMBER
- OVERRIDE_OF_SIGNATURE_CAPTURE=13045 - 3045 ENTER=BYPASS E-SIGN CLEAR=RETRY E-SIGN
- 13045=13045 - 3045 ENTER=BYPASS E-SIGN CLEAR=RETRY E-SIGN
- EXPECTING_CARD_VALIDATION_NUMBER=13052 - 3052 ENTER CARD VALIDATION NUMBER
- 13052=13052 - 3052 ENTER CARD VALIDATION NUMBER
- MGR_OVERRIDE_SUCCESS=5000 - Manager Override Successful
- MRG_OVERRIDE_CANCEL=5001 - Manager Override Canceled
- OPR_OVERRIDE_SUCCESS=5002 - Operator Override Successful
- OPR_OVERRIDE_CANCEL=5003 - Operator Override Canceled
- VALID_ACCOUNT_ENTERED=5004 - Valid Account Entered
- ITEM_PROMPT_CANCELED=5005 - Item Weight, Price, or Qty Prompt Canceled
- ITEM_PROMPT_SATISFIED=5006 - Item Weight, Price, or Qty Prompt Satisfied

nextFile=usersubstates – This property is used to tell AEF the name of the next property file in the bundle chain.

Appsubstates.properties (GSA)

The appsubstates.properties file includes the following properties.

- ACCOUNT_NUMBER_ENTERED=9000 - ACCOUNT NUMBER ENTERED (State 51)
- ACCOUNT_NUMBER_NOT_ENTERED=9001 - ACCOUNT NUMBER NOT ENTERED (State 51)
- TENDER_APPROVED=9002 - TENDER APPROVED (State 6)
- TENDER_NOT_APPROVED=9003 - TENDER NOT APPROVED (State 6)
- PRINT_BYPASS_PRINT=9004 - PRINT BYPASS PRINT (State 94)
- PRINT_BYPASS_BYPASS=9005 - PRINT BYPASS BYPASS (State 94)
- ACCEPT_TENDER=9006 - ACCEPT TENDER (State 97)
- VOID_TENDER=9007 - VOID TENDER (State 97)
- INSERT_CHECK=9008 - INSERT CHECK (For face printing)
- PRICE_ENTERED=9009 - PRICE ENTERED (State 50)
- PRICE_NOT_ENTERED=9010 - PRICE NOT ENTERED (State 50)
- QUANTITY_ENTERED=9011 - QUANTITY ENTERED (State 49)
- QUANTITY_NOT_ENTERED=9012 - QUANTITY NOT ENTERED (State 49)
- ORIGINAL SALESPERSON_ENTERED=9013 - original salesperson entered (State 20)
- ORIGINAL SALESPERSON_NOT_ENTERED=9014 - original salesperson not entered (State 20)
- VOID_TOTAL=9015 – void total (Any State)

nextFile=usersubstates - This property is used to tell AEF the name of the next property file in the bundle chain.

Appsubstates.properties (SA)

The appsubstates.properties file includes the following properties.

- EXPECTING_ITEM=1001 - Expecting item

- CHANGE_OR_BAL_DUE=1002 - Change or balance due.
- OPERATOR_ON_ANOTHER_TERMINAL=1006 - B079 operator still active on another terminal.
- EXPECTING_MANAGER_OVERRIDE=1006
- EXPECTING_OPERATOR_OVERRIDE=1007
- SELECT_PROCEDURE=1008 - SELECT PROCEDURE OR ENTER ITEM
- TRANS_TOTALLED=1010 - Transaction totaled
- MGR_OVERRIDE_SUCCESS=5000 - Manager Override Successful
- MRG_OVERRIDE_CANCEL=5001 - Manager Override Canceled
- OPR_OVERRIDE_SUCCESS=5002 - Operator Override Successful
- OPR_OVERRIDE_CANCEL=5003 - Operator Override Canceled
- VALID_ACCOUNT_ENTERED=5004 - Valid Account Entered
- ITEM_PROMPT_CANCELED=5005 - Item Weight, Price, or Qty Prompt Canceled
- ITEM_PROMPT_SATISFIED=5006 - Item Weight, Price, or Qty Prompt Satisfied
- INVALID_KEY_SEQUENCE=10333 - B003 CHECK KEY SEQUENCE
- KEYED_DATA_OUT_OF_RANGE=10336 - B006 KEYED DATA OUT OF RANGE
- ITEM_LIMIT_CHECK=10339 - B009 ITEM LIMIT CHECK
- USE_LOOKUP_KEYS=10340 - B010 USE LOOKUP KEYS
- CLOSE_CASH_DRAWER=10343 - B013 CLOSE THE CASH DRAWER
- EXTENDED_PRICE_LIMIT_CHECK=10347 - B017 EXTENDED PRICE LIMIT CHECK
- TAKE_TOTAL=10350 - B020 TAKE TOTAL
- NO_QUANTITY_ON=10353 - B023 NO QUANTITY ON
- SELL_WITHOUT_REPEAT_KEY=10355 - B025 SELL ITEM WITHOUT REPEAT KEY
- ITEM_NOT_FOUND=10356 - B026 ITEM NOT FOUND
- NOT_FOR_SALE=10357 - B027 NOT FOR SALE
- CHECK_TENDER_VARIETY=10364 - B034 CHECK TENDER VARIETY
- CHECK_QUANTITY=10366 - B036 CHECK QUANTITY
- CHECK_PRICE=10367 - B037 CHECK PRICE
- CHECK_ITEM_DATA=10369 - B039 CHECK ITEM DATA
- CHECK_OVERRIDE_NUMBER=10389 - B059 CHECK OVERRIDE NUMBER
- CHANGE_AMOUNT_LIMIT=10390 - B060 CHANGE AMOUNT LIMIT CHECK
- KEYED_PRICE_LIMIT_CHECK=10392 - B062 KEYED PRICE LIMIT CHECK
- TENDER_AMOUNT_LIMIT=10390 - B063 TENDER AMOUNT LIMIT CHECK
- FULL_KEYING_SEQUENCE_REQUIRED=10398 - B068 FULL KEYING SEQUENCE REQUIRED
- AUTH_REQUIRED=10406 - B076 AUTHORIZATION REQUIRED
- INVALID_PASSWORD=10407 - B077 CHECK PASSWORD
- INVALID_OPERATOR=10408 - B078 CHECK OPERATOR NUMBER
- SIGNON_GUIDANCE=10410 - B080 KEY OPERATOR *,PASSWORD,SIGN-ON
- PROCEDURE_NOT_AVAILABLE_NOW=10424 - B094 PROCEDURE NOT AVAILABLE NOW
- POINTS_ENTRY_REQUIRES_CUSTOMER_NUMBER=10930 - B600 POINTS ENTRY REQUIRES CUSTOMER #
- CHECK_KEYED_CUSTOMER_NUMBER=10934 - B604 CHECK KEYED CUSTOMER NUMBER
- INVALID_DATE_ENTERED=11031 - B508 INVALID DATE ENTERED
- PRICE_NEEDED=11033 - B024 PRICE NEEDED
- QUANTITY_NEEDED=11034 - B029 QUANTITY NEEDED
- WEIGHT_NEEDED=11035 - B019 WEIGHT NEEDED
- REPEAT_SIGNOFF=11036 - REPEAT SIGN-OFF KEY OR PRESS CLEAR
- ENTER_STANDALONE=11040 - WAIT FOR RETRY OR ENTER STANDALONE
- RETRY_OR_BYPASS=11055 - ENTER TO RETRY OR CLEAR TO BYPASS
- CLOSED=11042 - Closed
- TRAINING_MODE=11045 - TRAINING MODE ON

- WAITING_ON_HOST_RESPONSE=11056 - WAITING ON HOST RESPONSE
- CHECK_CUSTOMER_NUMBER=11166 - B605 CHECK CUSTOMER NUMBER
- REMOVE_RECEIPT_FOR_CUSTOMER=30020 - REMOVE RECEIPT FOR CUSTOMER
- PROMPT_FOR_DUPLICATE_RECEIPT=1100002 - ENTER FOR DUPLICATE CLEAR TO BYPASS
- 10333=B003 - B003 CHECK KEY SEQUENCE
- 10338=B008 - B008 TRANSACION LIMIT CHECK

nextFile=usersubstates – This property is used to tell AEF the name of the next property file in the bundle chain.

Tender Map Bundle

The Tender Map bundle is used to map a tender name to an application specific tender type. This is done by concatenating the "tenderType" and "tenderVariety" fields of the "tender" XML event from the POS application. This concatenated key is then looked up in the classes.properties file to determine the Java classname to use to represent the tender line item.

Each entry in this file contains four fields which are described below.

1. **Key** – This is the value that will be used as keys in the classes.properties chain. It will be used to generate the tender line item by appending 'Tender' to the key to look up the tender line item class in classes.properties. It will be used to generate the 'add tender action' class key by prepending 'Add' and appending 'TenderAction', so that 'AddKeyTenderAction' is the resulting key for use in classes.properties. It will also be used to generate the 'void tender action' class key by prepending 'Void' and appending 'TenderAction', so that 'VoidKeyTenderAction' is the resulting key for use in classes.properties.
2. **Description** - should match the tender description as defined in the application. When the application provides its tender definitions, the tender type and variety listed in the property file may be overridden by the application data, so long as the description matches (ignoring case and punctuation differences). Note: This is the only field which should be translated!
3. **Type** - is the tender type as defined by the POS application.
4. **Variety** - is the tender variety as defined by the POS application.

The files that makeup the Tender Map bundle are tendermap.properties, apttendermap.properties, and usertendermap.properties.

Tendermap.properties

The tendermap.properties file includes the following properties.

nextFile=apptendermap – This property is used to tell AEF the name of the next property file in the bundle chain.

Apptendermap.properties (ACE)

The apptendermap.properties file includes the following properties.

- Cash="cash",1,1
- Check="check",2,1
- FoodStamps="foodstamps",3,1
- Credit="credit",4,1

- Visa="visa",4,1
- MasterCard="mastercard",4,2
- Discover="discover",4,3
- AmericanExpress="american express",4,4
- Amex="amex",4,4
- GiftCard="gift card",4,5

nextFile=usertendermap – This property is used to tell AEF the name of the next property file in the bundle chain.

Apptendermap.properties (GSA)

The apptendermap.properties file includes the following properties.

- Check="check",0,
- GiftCertificate="gift certificate",1,
- DueBill="due bill",2,
- TravelersCheck="travelers check",3,
- StoreCoupon="store coupon",4,
- VendorCoupon="vendor coupon",5,
- CreditPlanA="credit plan a",6,
- CreditPlanB="credit plan b",7,
- CreditPlanC="credit plan c",8,
- CreditPlanD="credit plan d",9,
- Visa="visa",10,
- MasterCard="mastercard",11,
- AmericanExpress="american express",12,
- Amex="AMEX",12,
- DinersClub="diners club",13,
- Miscellaneous="miscellaneous tender",14,
- Cash="cash",15,

nextFile=usertendermap – This property is used to tell AEF the name of the next property file in the bundle chain.

Apptendermap.properties (SA)

The apptendermap.properties file includes the following properties.

- Cash="cash",1,1
- Check="check",2,1
- FoodStamps="foodstamps",3,1
- Credit="credit",4,1
- CreditCard="credit card",4,1
- Visa="visa",4,1
- MasterCard="mastercard",4,1
- AmericanExpress="american express",4,1
- Amex="AMEX",4,1
- DinersClub="diners club",4,1

nextFile=usertendermap – This property is used to tell AEF the name of the next property file in the bundle chain.

Appendix C. Data Provider Properties

This appendix lists the property values maintained by the Data Provider. Each table corresponds to a category.

The Properties interfaces form an inheritance structure. Therefore some properties defined in a base interface (such as the `LineItemProperties` interface) apply to any of the child interfaces.

Note: Not all applications support all the listed properties.

CouponProperties

The constant `CouponProperties.CATEGORY` can be used for the category name. These properties will be valid for the last coupon in the transaction.

Property Name Constant	Property Name	Type	Notes
COUPON_TYPE	"couponType"	String	"Store" or "Manufacturer" coupon.
UNIT_PRICE	"unitPrice"	String	
DEAL_QUANTITY	"dealQuantity"	String	
WEIGHT	"weight"	String	
VALUE	"value"	String	The total value of the coupon. This depends on the pricing method, if the coupon applies to a weighted item, etc.
DEAL_PRICE	"dealPrice"	String	Only applies to "deal price" pricing method.
REDUCED_PRICE	"reducedPrice"	String	Only applies to "reduced price" pricing method.
MULTI_PRICING_GROUP	"groupID"	String	Only applies to group pricing methods.
AGE_RESTRICTION	"requiredAge"	String	If the coupon is age restricted, this is the required age (in years) to redeem the coupon.
REDUCES_FOODSTAMP_BALANCE	"reducesFoodstampBalanceDue"	String	"true" if this coupon reduces the foodstamp balance due.
COUPON_REPEAT_ALLOWED	"itemRepeatAllowed"	String	"true" if the operator can

Property Name Constant	Property Name	Type	Notes
			repeat the coupon in the transaction by hitting <ENTER>.
REDUCES_TAX_DUE	"reducesTaxDue"	String	"true" if the coupon reduces the tax balance due.
PRICING_METHOD	"pricingMethod"	String	Property value is the name of the coupon pricing method. Values may be "unit", "dealQuantity", "basePlus1", "groupThreshold", "reducedDeal", "increasedDeal", "alias".
MANUFACTURER_NUMBER	"manufacturerNumber"	String	Coupon mfg. number if coupon is a manufacturer coupon.
IS_VOIDED	"isVoided"	String	"true" if the coupon has been voided.
ITEM_TAXABLE	"itemTaxable"	String	"true" if the coupon is taxable. If the coupon is taxable, it does not reduce the tax balance due.
IS_DEPOSIT	"isDeposit"	String	"true" if the coupon represents a deposit item.
IS_REFUNDED	"isRefunded"	String	"true" if the coupon has been refunded.
ITEM_MODIFIER	"itemModifier"	String	Value represents application unique information.
RESTRICTED_PERIODS	"restrict"	Collection	Value is a Collection of TimeInterval elements which represents the period of time when the coupon either cannot be redeemed, or must be

Property Name Constant	Property Name	Type	Notes
			redeemed.

Table 19 Data Provider Coupon Properties

CustomerProperties

The constant [CustomerProperties](#).CATEGORY can be used for the category name. These properties will be valid for the last customer loyalty number entered for the transaction.

Property Name Constant	Property Name	Type	Notes
CUSTOMER_ID	"id"	String	Customer's loyalty number.
ID_TYPE	"idType"	String	Indicates if the customer number was the "primary" loyalty number, or an "alternate" loyalty number.
ID_EXPIRATION_DATE	"idExpirationDate"	String	
CUSTOMER_NAME	"name"	String	Format is unspecified.
CUSTOMER_EMAIL	"email"	String	Not currently supported.
TARGETED_COUPONS	"targetedCoupons"	Collection	Collection of Strings where each String in the collection is an item code of a targeted coupon.
POINTS_TOTALS	"pointsTotals"	Collection	Collection of PointsTotal objects. Each PointsTotal represents a loyalty points category.
POINTS_BALANCES	"pointsBalances"	Collection	Collection of String objects. Each element is the total number of points in a loyalty category.
MESSAGES	"messages"	Collection	Collection of String objects. Each element is a loyalty message targeted for the customer.

Table 20 Data Provider Customer Properties

DiscountProperties

The constant [DiscountProperties](#).CATEGORY can be used for the category name. These properties will be valid for the last line item or transaction discount entered for the transaction.

Property Name Constant	Property Name	Type	Notes
DISCOUNT_METHOD	"discountMethod"	String	Either "percent" or "allowance". An allowance is a fixed amount discount.
DISCOUNT_AMOUNT	"amount"	String	The net amount of the discount.
DISCOUNT_RATE	"rate"	String	For a percent discount, this is the percentage. Format is a percentage, for example, 15% is represented as "15.0".
DISCOUNT_REDUCE_TAX	"reducesTaxBalanceDue"	String	"true" if the discount causes the tax due to be decreased. The tax due is decreased when the tax is calculated after the discount has been applied.
IS_TRANSACTION_DISCOUNT	"isTransactionDiscount"	String	"true" if the discount applies to the entire transaction.
IS_VOIDED	"isVoided"	String	"true" if the discount has been voided.
DISCOUNT_TYPE	"discountType"	String	Application specific information.
DISCOUNT_REASON	"discountReason"	String	Application specific discount reason code.

Table 21 Data Provider Discount Properties

ItemSalesProperties

The constant [ItemSalesProperties](#).CATEGORY can be used for the category name. These properties will be valid for the last item entered for the transaction.

Property Name Constant	Property Name	Type	Notes
UNIT_PRICE	"unitPrice"	String	Applies if pricing method is "unit".
WEIGHT	"weight"	String	The item's weight if sold by weight.
EXTENDED_PRICE	"extendedPrice"	String	
ITEM_REPEAT_ALLOWED	"itemRepeatAllowed"	String	"true" if the operator can sell another of the same item using a "repeat" key sequence.
IS_RETURN	"isReturn"	String	"true" if the line item is an item return.
RETURN_REASON	"returnReason"	String	Application specific return reason code. Applies only if the line item is an item return.
REQUIRED_AGE	"requiredAge"	String	The number of years old the customer must be to purchase this item. If no value, the item is not age restricted.
RESTRICTED_PERIODS	"restrict"	Collection	Value is a Collection of TimeInterval elements which represents the period of time when the item cannot be purchased, or must be purchased.
FOOD_STAMP_ELIGIBLE	"foodstampEligible"	String	"true" if the item may be purchased with foodstamps.
LINKED_ITEM_ID	"linkedItemID"	String	Item code of item which is automatically sold when this item is sold.
LINKED_ITEM_ID_QUALIFIER	"linkedItemIDQualifier"	String	Application specific

Property Name Constant	Property Name	Type	Notes
			information about the linked item ID.
DEAL_QUANTITY	"dealQuantity"	String	The quantity involved in dealQuantity-pricing, or other pricing methods which require a quantity.
DEAL_PRICE	"dealPrice"	String	The price for a deal pricing method. For example, if a deal is 3 for \$1.00, the dealPrice is "1.00".
REDUCED_PRICE	"reducedPrice"	String	A threshold price used by certain pricing methods.
MULTI_PRICING_GROUP	"groupID"	String	Identifies the group that the item belongs to for multigroup pricing schemes.
PRICING_METHOD	"pricingMethod"	String	The pricing method. For IBM 4690 Supermarket Application, the values may be: "unit", "dealQuantity", "basePlus1", "groupThreshold", "reducedDeal", "increasedDeal", "alias".
IS_VOIDED	"isVoided"	String	"true" if the line item is an item void.
ITEM_TAXABLE	"itemTaxable"	String	"true" if the item is taxable.
IS_DEPOSIT	"isDeposit"	String	"true" if the item is a deposit item.
IS_REFUNDED	"isRefunded"	String	"true" if the item is a line item refund.
ENTERED_PRICE_USED	"enteredPriceUsed"	String	"true" if the operator entered the item price (because the item was a price required item, or because of a price override).

Property Name Constant	Property Name	Type	Notes
ORIGINAL_SALESPERSON	"originalSalesperson"	String	If the application can provide this information, it is the salesperson who originally rang up the item. Usually used for item returns.

Table 22 Data Provider ItemSales Properties

LineItemProperties

The properties in the [LineItemProperties](#) interface apply to all line items. The following properties apply to the last LineItem in the transaction.

Property Name Constant	Property Name	Type	Notes
ITEM_ID	"itemID"	String	The item identifier. May be UPC, velocity code, etc.
ITEM_ID_QUALIFIER	"weight"	String	Indicates the format of the item code (may be scanned, keyed). For IBM 4690 Supermarket Application, the values may be: ScannedItemCode, KeyedItemCode, ItemLookupKeyed, LinkedItemCode, WandedItemCode
QUANTITY	"quantity"	String	
DESCRIPTION	"description"	String	
IS_VOIDED	"isVoided"	String	"true" if the line item is voided.
IS_DEPOSIT	"isDeposit"	String	"true" if the line item is a deposit item.
IS_REFUNDED	"isRefunded"	String	"true" if the line item is refunded.
PRINT_LINES	"printLines"	Collection	Value is a Collection of String elements. Each string represents a print line on the receipt. Any unprintable characters have been filtered from the strings.
RAW_PRINT_LINES	"rawPrintLines"	Collection	Value is a Collection of String elements. Each string represents a print line on the receipt. Any unprintable characters (such as those with

Property Name Constant	Property Name	Type	Notes
			ASCII values below 32) are transformed as <code>[\nn]</code> where <i>nn</i> is the ASCII value of the character.

Table 23 Data Provider LineItem Properties

OperatorAuthorizationProperties

The constant [OperatorAuthorizationProperties](#).CATEGORY can be used for the category name. These properties will be valid for the operator who is currently logged onto the session.

Authorization Properties (Common)

The following table lists authorization properties common to two or more of the IBM POS applications (GSA, SA, and ACE).

Property Name Constant	Property Name	Type	Notes
ITEM_PRICE_CHANGE_ALLOWED	"itemPriceChangeAllowed"	String	"true" or "false"
LAYAWAY_ALLOWED	"layawayAllowed"	String	"true" or "false"
LOAN_ALLOWED	"loanAllowed"	String	"true" or "false"
OBTAIN_ITEM_MOVEMENT_ALLOWED	"obtainItemMovementAllowed"	String	"true" or "false"
OBTAIN_DEPT_TOTALS_ALLOWED	"deptTotalsReportAllowed"	String	"true" or "false"
OPERATOR_TRAINING_ALLOWED	"operatorTrainingAllowed"	String	"true" or "false"
REENTRY_OFFLINE_SALES_ALLOWED	"reentryOfflineSalesAllowed"	String	"true" or "false"
REGISTER_READOUT_ALLOWED	"registerReadoutAllowed"	String	"true" or "false"
RETURNS_ALLOWED	"returnsAllowed"	String	"true" or "false"
DISCOUNTS_ALLOWED	"discountsAllowed"	String	"true" or "false"
VOID_TRANSACTION_ALLOWED	"voidTransactionAllowed"	String	"true" or "false"
PRICE_VERIFICATION_ALLOWED	"priceVerifyAllowed"	String	"true" or "false"
TERMINAL_MONITOR_ALLOWED	"terminalMonitorAllowed"	String	"true" or "false"
TENDER_LISTING_ALLOWED	"tenderListingAllowed"	String	"true" or "false"

Table 24 Data Provider Authorization Properties (Common)

Authorization Properties (GSA)

The following table lists authorization properties related to the IBM POS application GSA.

Property Name Constant	Property Name	Type	Notes
PASSWORD_REQUIRED	"passwordRequired"	String	"true" or "false"
ALLOWANCE_ALLOWED	"allowanceAllowed"	String	"true" or "false"
CASH_TRANSACTION_ALLOWED	"cashTransactionAllowed"	String	"true" or "false"
CASH_DOC_ALLOWED	"cashDocumentTransactionAllowed"	String	"true" or "false"
CASH_SPEC_ALLOWED	"cashSpecialTransactionAllowed"	String	"true" or "false"
CHARGE_TYPE_6_ALLOWED	"chargeType6TransactionAllowed"	String	"true" or "false"
CHARGE_TYPE_7_ALLOWED	"chargeType7TransactionAllowed"	String	"true" or "false"
CHARGE_TYPE_8_ALLOWED	"chargeType8TransactionAllowed"	String	"true" or "false"
CHARGE_TYPE_9_ALLOWED	"chargeType9TransactionAll"	String	"true" or "false"

Property Name Constant	Property Name	Type	Notes
	owed"		
TENDER_REMOVAL_ALLOWED	"tenderRemovalAllowed"	String	"true" or "false"
COD_ALLOWED	"codTransactionAllowed"	String	"true" or "false"
DELAYED_PRICE_CHANGE_ALLOWED	"delayedPriceChangeAllowed"	String	"true" or "false"
REGISTER_RESET_ALLOWED	"registerResetAllowed"	String	"true" or "false"
SET_TRANSACTION_NUMBER_ALLOWED	"setTransactionNumberAllowed"	String	"true" or "false"
SUSPEND_TRANSACTION_ALLOWED	"suspendTransactionAllowed"	String	"true" or "false"
TPL_TRANSACTION_ALLOWED	"tplTransactionAllowed"	String	"true" or "false"
VOID_PREVIOUS_BY_LINE_ALLOWED	"voidPreviousItemAllowed"	String	"true" or "false"
VOID_PREVIOUS_TRANSACTION_ALLOWED	"voidPreviousTransactionAllowed"	String	"true" or "false"
QUERY_EXCHANGE_RATE_ALLOWED	"queryExchangeRateAllowed"	String	"true" or "false"
NO_SALE_ALLOWED	"nosaleOpenCashDrawerAllowed"	String	"true" or "false"
WITHDRAWALS_TRANSACTION_ALLOWED	"pickupAllowed"	String	"true" or "false"
CASH_COUNT_ALLOWED	"tenderCountAllowed"	String	"true" or "false"

Table 25 Data Provider Authorization Properties (GSA)

Authorization Properties (SA & ACE)

The following table lists authorization properties related to the IBM POS applications SA & ACE.

Property Name Constant	Property Name	Type	Notes
CHECKOUT_TRANSACTION_ALLOWED	"salesAllowed"	String	"true" or "false"
TENDER_CASHING_ALLOWED	"tenderCashingAllowed"	String	"true" or "false"
TENDER_EXCHANGE_ALLOWED	"tenderExchangeAllowed"	String	"true" or "false"
PICKUP_TRANSACTION_ALLOWED	"pickupAllowed"	String	"true" or "false"
TERMINAL_TRANSFER_ALLOWED	"terminalTransferAllowed"	String	"true" or "false"
TENDER_COUNT_ALLOWED	"tenderCountAllowed"	String	"true" or "false"
RETURN_ITEM_ALLOWED	"returnItemAllowed"	String	"true" or "false"
WIC_TRANSACTION_ALLOWED	"wicTransactionAllowed"	String	"true" or "false"
USER_NON_SALES_1_ALLOWED	"userNonsales1Allowed"	String	"true" or "false"
USER_NON_SALES_2_ALLOWED	"userNonsales2Allowed"	String	"true" or "false"
NO_SALE_OPEN_CASH_DRAWER_ALLOWED	"nosaleOpenCashDrawerAllowed"	String	"true" or "false"
NO_SALE_TENDER_REMOVAL_ALLOWED	"nosaleTenderRemovalAllowed"	String	"true" or "false"
NO_SALE_TILL_EXCHANGE_ALLOWED	"nosaleTillExchangeAllowed"	String	"true" or "false"
NO_SALE_TENDER_VERIFY_ALLOWED	"nosaleTenderVerifyAllowed"	String	"true" or "false"
NO_SALE_TILL_REPORT_ALLOWED	"nosaleTenderVerifyAllowed"	String	"true" or "false"

Property Name Constant	Property Name	Type	Notes
NO_SALE_TILL_REPORT_ALLOWED	"nosaleTillReportAllowed"	String	"true" or "false"
NO_SALE_PRICE_VERIFY_ALLOWED	"nosalePriceVerifyAllowed"	String	"true" or "false"
REFUNDS_ALLOWED	"refundsAllowed"	String	"true" or "false"
MISC_ITEM_PAYOUTS_ALLOWED	"miscItemPayoutAllowed"	String	"true" or "false"
IMMEDIATE_IR_CHANGES_ALLOWED	"immediateItemChangeAllowed"	String	"true" or "false"
DELAYED_IR_CHANGES_ALLOWED	"delayedItemChangeAllowed"	String	"true" or "false"
MORE_THAN_PRICE_CHANGES_ALLOWED	"moreThanPriceChangesAllowed"	String	"true" or "false"
MANAGERS_PROCEDURES_ALLOWED	"managersProceduresAllowed"	String	"true" or "false"
REPRINT_TENDER_RECEIPT_ALLOWED	"reprintTenderReceiptAllowed"	String	"true" or "false"
USER_FUNCTION_1_ALLOWED	"userFunction1Allowed"	String	"true" or "false"
FRONT_END_CASHIER_ALLOWED	"frontEndCashierAllowed"	String	"true" or "false"
DEPT_TOTALS_REPORT_ALLOWED	"deptTotalsReportAllowed"	String	"true" or "false"

Table 26 Data Provider Authorization Properties (SA & ACE)

OptionsProperties

The constant [OptionsProperties](#).CATEGORY can be used for the category name. These properties will reflect the latest personalization options loaded by the POS application.

Property Name Constant	Property Name	Type	Notes
STORE_DEFINITION	"store"	StoreDefinition	The StoreDefinition contains information such as store number, name, address, and phone numbers.
TENDER_DEFINITION	"tenderDefinition"	Collection	Value is a Collection of TenderDefinition elements. Each element contains the POS application's definition of a tender.
PRICE_OVERRIDE_REASON	"priceOverrideReason"	Collection	Value is a Collection of ReasonCode elements. This collection represents the price override reason codes configured in the POS application.
REFUND_REASON	"refundReason"	Collection	Value is a Collection of ReasonCode elements. This collection represents the refund reason codes configured in the POS application.
VOID_REASON	"voidReason"	Collection	Value is a Collection of ReasonCode elements. This collection represents the void reason codes configured in the POS application.
TARE_CODE	"tareCode"	Collection	Value is a Collection of

Property Name Constant	Property Name	Type	Notes
			TareCode elements. This collection represents the tare codes configured in the POS application.
VAT_TAX_CODE	"vatTaxCode"	Collection	Value is a Collection of TaxCode elements. This collection represents the VAT tax codes configured in the POS application.
ALTERNATE_TAX_CODE	"alternateTaxCode"	Collection	Value is a Collection of TaxCode elements. This collection represents the tax codes configured in the POS application which represent alternate tax tables.
MANUAL_TAX_CODE	"manualTaxCode"	Collection	Value is a Collection of TaxCode elements. This collection represents the tax codes configured in the POS application which represent manual tax amount entries.
NO_TAX_CODE	"noTaxCode"	Collection	Value is a Collection of TaxCode elements. This collection represents the tax codes configured in the POS application which represent zero tax entries.
TRANSACTION_DISCOUNT_REASON_CODE	"transactionDiscountReason"	Collection	Value is a Collection of TransactionDiscountReasonC

Property Name Constant	Property Name	Type	Notes
			ode elements. This collection represents the transaction discount reason codes configured in the POS application.
ITEM_DISCOUNT_REASON_CODE	"itemDiscountReason"	Collection	Value is a Collection of DiscountReasonCode elements. This collection represents the item discount reason codes configured in the POS application.
DEPARTMENT_DEFINITION	"departmentDefinition"	Collection	Value is a Collection of DepartmentDefinition elements. This collection represents the departments configured in the POS application.
TERMINAL_OPTIONS	"terminalOptions"	Collection	Value is a Collection of TerminalOptions elements. This collection represents terminal options configured in the POS application.
STORE_OPTIONS	"storeOptions"	Collection	Value is a Collection of StoreOptions elements. This collection represents terminal options configured in the POS application.

Table 27 Data Provider Options Properties

PointsProperties

The constant [PointsProperties](#).CATEGORY can be used for the category name. These properties will reflect the latest loyalty points which were rewarded or redeemed within the the POS application transaction.

Property Name Constant	Property Name	Type	Notes
ID	"id"	String	Identifier which identifies the points. For IBM applications such as SA and ACE, this is the points item code.
ID_QUALIFIER	"idQualifier"	String	Indicates format of the points identifier.
TYPE	"type"	String	Refers to the loyalty points category.
TOTAL	"points"	String	The number of points rewarded or redeemed.
POINTS_REDEEMED	"redeemed"	String	"true" if the points were redeemed in the transaction, "false" if they were awarded.
POINTS_VOIDED	"isVoided"	String	"true" if the points were voided in the transaction.
ADDITIONAL_POINTS_TOTALS	"pointsTotals"	Collection	Value is a Collection of PointsTotal elements. Each element represents the new points total for each points category.

Table 28 Data Provider Points Properties

POSDeviceProperties

The constant [POSDeviceProperties](#).CATEGORY can be used for the category name. These properties will reflect POS I/O device and I/O processor attributes. Note that the appropriate devices must be configured as redirected for the property values to be set properly.

Property Name Constant	Property Name	Type	Notes
ANPROMPT_LINE1	"ANPROMPT_LINE"	String	Contains the contents of the first line of the operator line display.
ANPROMPT_LINE2	"ANPROMPT_LINE2"	String	Contains the contents of the second line of the operator line display.
OPERATOR_PROMPT_LINE1	"ANPROMPT_LINE"	String	Contains the contents of the first line of the operator line display.
OPERATOR_PROMPT_LINE1	"ANPROMPT_LINE2"	String	Contains the contents of the second line of the operator line display.
CUST_PROMPT_LINE1	"CUST_PROMPT_LINE"	String	Contains the contents of the first line of the customer line display.
CUST_PROMPT_LINE2	"CUST_PROMPT_LINE2"	String	Contains the contents of the second line of the operator line display.
OPERATOR_DISPLAY	"OPERATOR_DISPLAY"	String	Indicates which line display is being used by the POS application as the operator display. Possible values are "LineDisplay1", "LineDisplay2", "LineDisplay3".
CUSTOMER_DISPLAY	"CUSTOMER_DISPLAY"	String	Indicates which line display is being used by the POS application as the customer display. Possible values are

Property Name Constant	Property Name	Type	Notes
			"LineDisplay1", "LineDisplay2", "LineDisplay3".
POS_SUB_STATE	"subState"	String	A numeric value which indicates the current substate of the POS application.
POS_STATE	"POS_STATE"	String	A numeric value which indicates the current IO processor state of the POS application.
CHANGE_TO_CURRENT	"CHANGE_TO_CURRENT"	String	"true" indicates that the last POS_STATE change was a transition from a state to the same state. Note that this is a specific definition in the application state table. The function code must be defined to "CHANGE TO CURRENT". State changes from a state to the same state for other reasons will not cause this flag to be set.
SYSTEM_BUSY	"SYSTEM_BUSY"	String	"true" indicates that the IO processor input queue is locked. No keyboard or scanner input is accepted.
PAPER_CUT	"paperCut"	String	"true" indicates that the POS application has sent a paper-cut command to the receipt printer.
PRINT_LINE	"printLine"	String	A string containing the last print line sent to the receipt printer.
PRINT_LINE_ARRAY	"PRINT_LINE_ARRAY"	ArrayList	Value is an

Property Name Constant	Property Name	Type	Notes
			ArrayList of String elements. Each String element is a cash receipt print line for the current transaction.
PRINT_LINE_FEEDS	"lineFeeds"	String	A numeric value indicating the last number of line feeds sent to the receipt printer.
APP_WILL_PROVIDE_DATA	"APP_WILL_PROVIDE_DATA"	String	<p>"true" indicates that the POS application will provide the cash receipt data via XML events. "false" indicates that the cash receipt data will come from a Device Accessor printer hook. The problem with using the printer hook approach is the delay in receiving print lines from applications which use clean receipt or post printing.</p> <p>The value for this property is provided by the POS application.</p>
SCALE_WEIGHT_VALUE	"scaleWeightValue"	String	A numeric value showing the weight of the item currently on the scale.
SCALE_WEIGHT_UNIT	"scaleWeightUnit"	String	Indicates the units of measure of the scale weight.
SCALE_WEIGHT_LABELS	"scaleWeightLabels"	String	Additional scale display text as required by Weights and Measures.
KEYLOCK_IS_SUPERVISOR	"KEYLOCK_IS_SUPERVISOR"	String	"true" indicates

Property Name Constant	Property Name	Type	Notes
	OR		that the keylock is in the “manager” position. “false” indicates the keylock is in “normal” position.
CASH_DRAWER_OPEN	“CASH_DRAWER_OPEN”	String	“true” indicates the cash drawer is open. “false” indicates the drawer is closed.
MSR_TRACK_1	“MSR_TRACK_1”	String	The value of the track 1 data of the last card swiped through the magnetic stripe reader.
MSR_TRACK_2	“MSR_TRACK_2”	String	The value of the track 2 data of the last card swiped through the magnetic stripe reader.
MSR_TRACK_3	“MSR_TRACK_3”	String	The value of the track 3 data of the last card swiped through the magnetic stripe reader.
PRINTER_COVER_OPEN	“PRINTER_COVER_OPEN”	String	“true” indicates the cash receipt printer cover is open. “false” indicates the cover is closed.
PRINTER_DOC_INSERT	“PRINTER_DOC_INSERT”	String	“true” indicates that there is a document inserted in the printer document insert station. “false” indicates there is no document in the document insert printer station.
PRINTER_OUT_OF_RECEIPT_PAPER	“PRINTER_OUT_OF_RECEIPT_PAPER”	String	“true” indicates the printer is out of receipt paper.

Table 29 Data Provider POSDevice Properties

StoreOptionsProperties

The constant [StoreOptionsProperties](#).CATEGORY can be used for the category name. These properties represent the last “store options” configuration values loaded by the POS application.

Property Name Constant	Property Name	Type	Notes
NO_SALE_PRICE_VERIFY_IN_TRANSACTION_ALLOWED	"priceVerifyInsideTransactionAllowed"	String	“true” indicates that the function is allowed. “false” indicates the function is not allowed. (SA, ACE)
MANAGERS_KEY_NEEDED	"overrideRequiresManagerKey"	String	“true” indicates that the POS application is configured to require a manager’s key for a manager’s override. (SA, ACE)
MAXIMUM_TRANSACTION_SIZE	"maximumTransactionSize"	String	The maximum number of line items allowed in a transaction as configured by the POS application. (SA, ACE)
TRANSACTION_WARNING_SIZE	"transactionWarningSize"	String	The number of line items configured by the POS application which will generate a transaction size warning. (SA, ACE)
FOOD_STAMPS_ALLOWED	"foodstampsAllowed"	String	“true” indicates that the function is allowed. “false” indicates the function is not allowed. (SA, ACE)
FOOD_STAMPS_ONLY_ALLOWED	"onlyFoodstampsAllowedAfterFoodstampTotal"	String	“true” indicates that the POS application is configured to accept only foodstamps after a foodstamp total is taken. (SA, ACE)

Property Name Constant	Property Name	Type	Notes
MAX_SUSPENDED_TRANSACTION	"maximumSuspendedTransactions"	String	The maximum number of suspended transaction as configured in the POS application. (SA, ACE)
CUSTOMER_FUNCTION_CODE	"customerFunctionCode"	String	They keyboard function code which is configured by the POS application for entry of the customer loyalty number. (SA-EM, ACE)
SUSPEND_TRANSACTION_ALLOWED	"suspendTransactionAllowed"	String	"true" indicates that the function is allowed. "false" indicates the function is not allowed. (SA, ACE)
WIC_TENDER_ONLY_IN_WIC_TRANSACTION	"WICTenderOnlyInWICTransaction"	String	"true" indicates that the POS application is configured to allow only WIC tenders within WIC transactions. (SA, ACE)
WEIGHT_DECIMAL_PLACES	"weightInputDecimalPlaces"	String	The numeric value configured in the POS application for the number of decimal places when entering a weight. (SA, ACE)
FOREIGN_TENDER_SUPPORTED	"foreignTenderAllowed"	String	"true" indicates that the POS application is configured to accept foreign tenders. (SA, ACE)
ALPHA_STATE_INPUT_ALLOWED	"alphaStateInputAllowed"	String	"true" indicates that the POS application is configured to allow alphabetic input of states. (ACE)

Property Name Constant	Property Name	Type	Notes
ALPHA_DRIVERS_LICENSE_INPUT_ALLOWED	"alphaDriversLicenseInputAllowed"	String	"true" indicates that the POS application is configured to allow alphabetic input of states for driver's license entry. (ACE)
MULTIPLE_CASH_DRAWER_SUPPORT	"multipleCashDrawerSupport"	String	"true" indicates that the POS application is configured to support multiple cash drawers. (ACE)
OTR_ENABLED	"openTransReportEnabled"	String	"true" indicates that the open transaction report has been enabled by the POS application. (ACE)
OTR_PRINT_ENABLED	"openTransPrintEnabled"	String	"true" indicates that printing the open transaction report has been enabled by the POS application. (ACE)
OTR_REFRESH_ENABLED	"openTransRefreshEnabled"	String	"true" indicates that refreshing the open transaction report has been enabled by the POS application. (ACE)
OTR_FUNCTIONS_ENABLED	"openTransFunctionsEnabled"	String	"true" indicates that the open transaction report functions have been enabled in the POS application. (ACE)
COUPON_MULTIPLIER_ENABLED	"couponMultiplierEnabled"	String	"true" indicates that the function has been configured in the POS application. (ACE)
EATIN_TAKEOUT_PROMPT_ENABLED	"eatinTakeoutPromptEnabled"	String	"true" indicates that the function has been

Property Name Constant	Property Name	Type	Notes
			configured in the POS application. (ACE)
VOLUME_INPUT_DECIMAL_PLACES	"volumeInputDecimalPlaces"	String	The numeric value configured in the application for the number of decimal places used for volume input. (ACE)
VOLUME_UNIT_PRICE_DECIMAL_PLACES	"volumeUnitPriceDecimalPlaces"	String	The numeric value configured in the application for the number of decimal places used for volume unit prices. (ACE)
GIFT_RECEIPT_PRINTING_ENABLED	"enableGiftReceiptPrinting"	String	"true" indicates that the function is configured. (ACE)
SCAN_MANAGER_ID_REQUIRED	"scanManagerID"	String	"true" indicates that the POS application is configured to require the manager ID to be scanned for manager overrides. (ACE)
WICEBT_ID	"WICEBTid"	String	(ACE)
TRANSACTION_DEFINITION	"transactionDefinition"	Collection	A Collection of TransactionDefinition elements. Each element represents the configuration settings for a different transaction type. (GSA)

Table 30 Data Provider StoreOptions Properties

TenderProperties

The constant [TenderProperties](#).CATEGORY can be used for the category name.
These properties represent the attributes of the last tender line item in the transaction.

Property Name Constant	Property Name	Type	Notes
TENDER_TYPE	"tenderType"	String	The type of the tender as defined by the POS application. For the IBM POS applications, the tender type is an integer.
TENDER_VARIETY	"tenderVariety"	String	The variety of tender (as defined by the POS application. (SA, ACE)
AMOUNT	"amount"	String	The tender amount (without currency symbols).
ACCOUNT_NUMBER	"accountNumber"	String	The tender account number if applicable.
EXPIRATION_DATE	"expirationDate"	String	The tender expiration date if applicable.
FEE	"fee"	String	The tender fee (without currency symbol) if the POS application applied a fee for taking the tender.

Table 31 Data Provider Tender Properties

TerminalOptionsProperties

The constant [TerminalOptionsProperties](#).CATEGORY can be used for the category name. These properties represent the last "terminal options" configuration values loaded by the POS application.

Property Name Constant	Property Name	Type	Notes
NO_SALE_ALLOWED	"noSaleTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
CASH_TRANSACTION_ALLOWED	"cashTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
LAYAWAY_ALLOWED	"layawayTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
RETURNS_ALLOWED	"itemReturnTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
VOID_TRANSACTION_ALLOWED	"voidTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
TAX_CODE_REQUIRED	"taxCodeRequired"	String	"true" indicates the POS application is configured to require a tax code when changing the taxes. (GSA)
OPERATOR_ID_REQUIRED	"operatorIDRequired"	String	(GSA)
CASH_SPEC_ALLOWED	"cashSpecialTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
CASH_DOC_ALLOWED	"cashDocumentTransactionAllowed"	String	"true" indicates the function is allowed, "false"

Property Name Constant	Property Name	Type	Notes
			indicates the function is not allowed. (GSA)
CHARGE_PLAN_A_ALLOWED	"chargePlanATransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
CHARGE_PLAN_B_ALLOWED	"chargePlanBTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
CHARGE_PLAN_C_ALLOWED	"chargePlanCTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
CHARGE_PLAN_D_ALLOWED	"chargePlanDTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
COD_ALLOWED	"codTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
SEND_ALLOWED	"sendTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
PAYMENTS_ALLOWED	"paymentsAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
ALLOWANCE_ALLOWED	"allowanceAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)
DISCOUNTS_ALLOWED	"discountsAllowed"	String	"true" indicates the function is allowed, "false"

Property Name Constant	Property Name	Type	Notes
			indicates the function is not allowed. (GSA)
ORIGINAL_SALES_PERSON_REQUIRED	"originalSalesPersonRequired"	String	"true" indicates the application is configured to require the entry of the original salesperson. (GSA)
TERMS_OF_SALE_REQUIRED	"termsOfSaleRequired"	String	(GSA)
SUSPEND_TRANSACTION_ALLOWED	"suspendTransactionAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA)

Table 32 Data Provider TerminalOptions Properties

TransactionStatusProperties

The constant [TransactionStatusProperties](#).CATEGORY can be used for the category name. These properties represent information about the POS transaction in progress.

Property Name Constant	Property Name	Type	Notes
ID	"id"	String	The transaction identifier (number).
DATE	"date"	String	The date the transaction was initiated.
TIME	"time"	String	The time the transaction was initiated.
TRANSACTION_CATEGORY	"category"	String	Should be either "sales" or "nonsales" depending on the transaction type.
TRANSACTION_TYPE	"type"	String	The transaction type. Expected values from GSA, SA, & ACE would be: "noSale" "cash" or "regularSale" which are equivalent "cashSpecial" "cashDocument" "cod" "layaway" "chargePlanA" "chargePlanB" "chargePlanC" "chargePlanD" "loan" "pickup" (aka withdrawal) "tenderCount" "totalsReadoutReset" "itemMovementReport" "itemPriceChange" "setTransactionNumber" "offlineReentry" "training" "voidPreviousTransaction"

Property Name Constant	Property Name	Type	Notes
			"voidPreviousByLinItem" "tenderListingReport" "tenderRemoval" "terminalMonitor" "queryExchangeRate" "tenderCashing" "tenderExchange" "priceVerify" "terminalTransfer" "terminalProgramLoad" "itemReturn" "wic" "reprintPartialReceipt" "reprintTenderReceipt" "EBTBalanceInquiry" "valueCardBalanceInquiry" "departmentTotalReport" "layawayPayment" "layawayCancel" "priceChange" "tenderFeeRefund" "suspendedTransactionReport" "modifyDepartmentPresets" "openTransactionReport"
TAX_REASON	"taxReason"	String	The tax code or reason code for the last tax change in the transaction.
TAX_TYPE	"taxType"	String	(GSA) – Set to the type of taxcode. (SA) – Set to "taxExempt" if the taxReason is a taxExempt discount (ACE) – Set to "taxExempt" if the taxReason is a

Property Name Constant	Property Name	Type	Notes
			discount group which exempts all tax plans. Set to "alternateTax" if the taxReason is a discount group which exempts some tax plans.
TAX_VOIDED	"isVoided"	String	Set to "true" if the last tax change was voided.
RETURN_PAYMENTS	"returnPayments"	String	"true" indicates a layaway transaction for returning layaway payments was started. (GSA)
CANCEL_ALL_ITEMS	"cancelAllItems"	String	"true" indicates a layaway transaction to cancel all layaway items was started. (GSA)
MODIFIER	"modifier"	String	Additional transaction information which may be application specific. For example : "return" "send" "vat" "display"
ITEM_ALLOWANCE_ALLOWED	"itemAllowanceAllowed"	String	"true" indicates that item allowance is now valid in the transaction. (GSA)
ITEM_DISCOUNT_ALLOWED	"itemDiscountAllowed"	String	"true" indicates that item discount is now valid in the transaction. (GSA, ACE)
VOID_LINE_ITEM_ALLOWED	"itemVoidAllowed"	String	"true" indicates the void line item function is valid in the transaction. (GSA, SA, ACE)
TRANSACTION_DISCOUNT_ALLOWED	"transactionDiscountAllowed"	String	"true" indicates the transaction

Property Name Constant	Property Name	Type	Notes
			discount function is valid in the transaction. (GSA, SA, ACE)
VOID_TRANSACTION_DISCOUNT_ALLOWED	"voidTransactionDiscountAllowed"	String	"true" indicates the function is allowed, "false" indicates the function is not allowed. (GSA, SA, ACE)
LAST_CREDIT_APPROVED	"lastCreditApproved"	String	Set to "true" and then back to "false" whenever a credit tender is approved.
ITEM_SALE_IN_PROGRESS	"itemSaleInProgress"	String	Set to "true" when a line item is detected by the AEF, and set to "false" when the AEF receives the Unit of Work message from the POS application.
ORIGINAL_SALES_PERSON_REQUIRED	"originalSalesPersonRequired"	String	"true" indicates the application is configured to require the entry of the original salesperson. (GSA)

Table 33 Data Provider TransactionStatus Properties

TransactionTotalsProperties

The constant [TransactionTotalsProperties](#).CATEGORY can be used for the category name. These properties reflect the latest running transaction totals for the POS transaction in progress.

Property Name Constant	Property Name	Type	Notes
TOTAL	"total"	String	The total amount of the transaction.
SUB_TOTAL	"subTotal"	String	The total transaction amount less any taxes.
TOTAL_SAVINGS	"totalSavings"	String	The total loyalty savings in the transaction.
TAX	"tax"	String	The sum of all taxes in the transaction.
AMOUNT_DUE	"balanceDue"	String	The remaining balance due to satisfy the transaction.
CHANGE_DUE	"changeDue"	String	The amount of change due to the customer.
FOOD_STAMP_TOTAL	"foodStampTotal"	String	The total transaction amount which are eligible for food stamps.
FOOD_STAMP_BALANCE	"foodStampBalance"	String	The remaining food stamps amount due in the transaction.
FOOD_STAMP_CHANGE	"foodStampChange"	String	The amount of change in food stamps due the customer.
TENDER_EXCHANGED	"tenderExchanged"	String	The total tender amount (of any type) accepted for this transaction.
COUPON_TOTAL	"totalCouponAmount"	String	The total amount of coupons tendered in this transaction.
TOTAL_ITEMS	"totalItems"	String	The total number of items sold in this transaction.
TOTAL_COUPONS	"totalCouponCount"	String	The total number of coupons

Property Name Constant	Property Name	Type	Notes
			tendered in this transaction.
LAYAWAY_BALANCE_DUE	"layawayBalanceDue"	String	The remaining balance on the layaway account. (GSA)
LAYAWAY_FEE	"layawayFee"	String	For layaway transaction type, set to the layaway fee (if any). For layawayCancel transaction type, set to a negative layaway fee if the fee is refundable. (GSA)
LAYAWAY_DEPOSIT	"layawayDeposit"	String	For layaway transaction type, set to the layaway deposit amount (if any). For layawayCancel transaction type, set to a negative layaway deposit if the deposit is refundable. For a layawayPayment transaction type, set to the layaway deposit amount (if any). (GSA)
FOREIGN_BALANCE_DUE	"foreignBalanceDue"	String	The remaining balance translated to the foreign currency as configured in the application. Should be included whenever a foreign total is taken. (SA,ACE)

Table 34 Data Provider TransactionTotals Properties

WorkstationStatusProperties

The constant [WorkstationStatusProperties](#).CATEGORY can be used for the category name. These properties reflect various attributes of the virtual or real register.

Property Name Constant	Property Name	Type	Notes
TERMINAL_STATUS	"terminalStatus"	String	Values may be WorkstationStatusEvent.DISABLE_TERMINAL_ACKNOWLEDGED, or WorkstationStatusEvent.INPUT_SEQUENCE_CLEARED.
TERMINAL_NUMBER	"TERMINAL_NUMBER"	String	Set to the terminal number.
TRANSACTION_NUMBER	"TRANSACTION_NUMBER"	String	Set to the transaction number of the transaction in progress.
SALES_TRANSACTION_IN_PROGRESS	"SALES_TRANSACTION_IN_PROGRESS"	String	Set to "true" while a sales transaction is in progress.
TRANSACTION_IN_PROGRESS	"TRANSACTION_IN_PROGRESS"	String	Set to "true" while a transaction of any type is in progress.
REENTRY_OFFLINE_TRANSACTION	"REENTRY_MODE"	String	Set to "true" while the register is in offline reentry mode. (GSA)
TRAINING_MODE	"TRAINING_MODE"	String	Set to "true" while the register is in training mode.
OFFLINE_MODE	"OFFLINE_MODE"	String	Set to "true" while the register is offline from its controller. Note, a virtual CSS session will never be offline.
MESSAGE_PENDING	"MESSAGE_PENDING"	String	Set to "true" if there is a message pending for the register. On the IBM 4690 OS platform, this is a system message from the OS.
QUEUE_LOCKED	"QUEUE_LOCKED"	String	Set to "true" if the

Property Name Constant	Property Name	Type	Notes
			register's input queue is locked. If the queue is locked, the register cannot accept keyboard or scanner input.
UNIT_OF_WORK	"unitOfWork"	String	Set to "true" and then to "false" when the POS application has completed processing some input and returned to "idle" state.
OPTIONS_LOADING_IN_PROGRESS	"OPTIONS_LOADING_IN_PROGRESS"	String	Set to "true" while the register is loading configuration options.
SIGNON_STATUS	"signonStatus"	String	Indicates whether an operator is signed onto the register. Values are "true", "false", or "secureMode".
TERMINAL_DISABLED	"terminalDisabled"	String	Set to "true" while the terminal has been disabled.

Table 35 Data Provider WorkstationStatus Properties

Appendix D. POS Application XML Events

Even though these XML events are meant only to provide data between the IBM point of sale applications and the AEF, they are documented in anticipation that there may be user extensions which require user data to be included in the events.

Operator Events

signOn					
Sent	When operator logs onto the POS application.				
XML Format					
<pre><signOn operatorID="xxxx" operatorName="yyyyyy yyyy"> <authorization auth_prop1="true" auth_prop2="false" /> </signOn></pre>					
operatorID	String	The operator id.			
operatorName	String	The actual name of the operator .			
authorization	Element	Authorization properties are listed as attributes.			
		<table><tr><td>auth_propN</td><td>true/false</td></tr></table>	auth_propN	true/false	<p>Each authorization property is listed by name in the signon event, along with a boolean indicating whether the operator is authorized to perform the function represented by the property name. Authorization properties for AEF enabled IBM applications are provided in the interface class OperatorAuthorizationProperties. See the following table for a list of the authorization property names for each application.</p>
auth_propN	true/false				

Table 36 <signOn> Event

This table lists the operator authorization attributes for each application which may be expected in the <signOn> event.

Operator Authorization Property Name	GSA	SA	ACE
tenderCountAllowed	X (known as cash count)	X	X
cashTransactionAllowed	X		
itemPriceChangeAllowed	X		
layawayAllowed	X		
loanAllowed	X	X	X
nosaleOpenCashDrawerAllowed	X	X	X
deptTotalsReportAllowed	X		X
obtainItemMovementAllowed	X		
operatorTrainingAllowed	X	X	X
reentryOfflineSalesAllowed	X		
registerReadoutAllowed	X		
returnsAllowed	X	X (known as refunds)	X (known as refunds)

Operator Authorization Property Name	GSA	SA	ACE
voidTransactionAllowed	X		
pickupAllowed	X (known as withdrawals)	X	X
allowanceAllowed	X		
cashDocumentTransactionAllowed	X		
cashSpecialTransactionAllowed	X		
chargeType6TransactionAllowed	X		
chargeType7TransactionAllowed	X		
chargeType8TransactionAllowed	X		
chargeType9TransactionAllowed	X		
codTransactionAllowed	X		
delayedPriceChangeAllowed	X		
discountsAllowed	X	X	X
priceVerifyAllowed	X		
passwordRequired	X		
queryExchangeRateAllowed	X		
registerResetAllowed	X		
setTransactionNumberAllowed	X		
suspendTransactionAllowed	X		
tenderListingAllowed	X	X	X
tenderRemovalAllowed	X		
terminalMonitorAllowed	X	X	X
tplTransactionAllowed	X		
voidPreviousItemAllowed	X		
voidPreviousTransactionAllowed	X		
salesAllowed		X	X
delayedItemChangeAllowed		X	X
deptTotalsReportAllowed		X	X
frontEndCashierAllowed		X	X
immediateItemChangeAllowed		X	X
managersProceduresAllowed		X	X
miscItemPayoutAllowed		X	X
moreThanPriceChangesAllowed		X	X
nosalePriceVerifyAllowed		X	X
nosaleTenderRemovalAllowed		X	X
nosaleTenderVerifyAllowed		X	X
nosaleTillExchangeAllowed		X	X
nosaleTillReportAllowed		X	X
operatorTrainingAllowed		X	X
reprintTenderReceiptAllowed		X	X
returnItemAllowed		X	X
tenderCashingAllowed		X	X
tenderExchangeAllowed		X	X
terminalTransferAllowed		X	X
userFunction1Allowed		X	X
userNonsales1Allowed		X	X
userNonsales2Allowed		X	X
wicTransactionAllowed		X	X

Table 37 Operator Authorization Attributes

signOff		
Sent	When operator logs off the POS application.	
XML Format		
<signOff operatorID="xxxx" />		
operatorID	String	The operator id.

Table 38 <signOff> Event

lock		
Sent	When operator locks the workstation, or enters secured mode.	
XML Format		
<pre><lock operatorID="xxxx" /></pre>		
operatorID	String	The operator id.

Table 39 <lock> Event

unlock		
Sent	When operator unlocks the workstation by entering password.	
XML Format		
<pre><unlock operatorID="xxxx" /></pre>		
operatorID	String	The operator id.

Table 40 <unlock> Event

Customer Event

customer	
Sent	When the POS application has identified a loyalty customer (possibly by scanning, or swiping a loyalty card).
XML Format	
<pre><customer id="90071234566" loyaltyIDType="primary alternate" expirationDate="MMYY" name="Carina Lopez" email="carina@us.ibm.com"> <targetedCoupon couponCode="0312" /> . . . <pointsTotal type="primary" description="Primary Points Total" points="0" /> <pointsBalance type="primary" description="Primary Points Balance" points="0" /> <message> VALUED CUSTOMER Carina Lopez </message> </customer></pre>	
<p>Notes:</p> <p>The POS application may optionally include one or more targeted coupons for the customer. Each is identified by a coupon-code. The POS application may also include a message specifically for the customer.</p> <p>There may also be one or more <pointsBalance> and <pointsTotal> elements included.</p>	

id	String	The customer loyalty number.									
name	String	The customer's name in the application specific format.									
loyaltyIDType	String	Either 'primary' or 'alternate'. Example of an alternate loyalty id is the customer's phone number.									
expirationDate	String	An optional field which is provided when an expiration date was provided with the loyalty id.									
email	String	The customer's email address. Note: The Supermarket application leaves this field blank.									
targetedCoupon	Element	For each coupon targeted to the customer, include a <targetedCoupon> element with the following attributes: <table border="1"> <tr> <td>couponCode</td><td>String</td><td>The item code of the coupon.</td></tr> </table>	couponCode	String	The item code of the coupon.						
couponCode	String	The item code of the coupon.									
pointsTotal	Element	The total number of points (as determined by the POS application) for the customer. The type attribute will be used to distinguish points of different types. Each <pointsTotal> element will contain the following attributes: <table border="1"> <tr> <td>type</td><td>String</td><td>An indicator of the type of points. For example, may be a club.</td></tr> <tr> <td>description</td><td>String</td><td>A description of the points. For example, the club name.</td></tr> <tr> <td>points</td><td>String</td><td>The number of points of the specific type for the customer.</td></tr> </table>	type	String	An indicator of the type of points. For example, may be a club.	description	String	A description of the points. For example, the club name.	points	String	The number of points of the specific type for the customer.
type	String	An indicator of the type of points. For example, may be a club.									
description	String	A description of the points. For example, the club name.									
points	String	The number of points of the specific type for the customer.									
pointsBalance	Element	The remaining points after subtracting any points that have been redeemed by the customer. The type attribute will be used to distinguish points of different types. Each <pointsBalance> element will contain the following attributes: <table border="1"> <tr> <td>type</td><td>String</td><td>An indicator of the type of points. For example, may be a club.</td></tr> <tr> <td>description</td><td>String</td><td>A description of the points. For example, the club name.</td></tr> <tr> <td>points</td><td>String</td><td>The points balance of the specific type for the customer.</td></tr> </table>	type	String	An indicator of the type of points. For example, may be a club.	description	String	A description of the points. For example, the club name.	points	String	The points balance of the specific type for the customer.
type	String	An indicator of the type of points. For example, may be a club.									
description	String	A description of the points. For example, the club name.									
points	String	The points balance of the specific type for the customer.									
messages	String	The message displayed by the application when the customer is recognized.									
Optional attributes supported by the CustomerEvent but not provided by IBM 4690 applications.											
firstName	String	The customer's first name only									
lastName	String	The customer's last name only									
address1	String	The customer address line 1									
address2	String	The customer address line 2									

<i>city</i>	<i>String</i>	<i>The customer's city</i>
<i>state</i>	<i>String</i>	<i>The customer's state</i>
<i>zip</i>	<i>String</i>	<i>The customer zip code</i>
<i>phone</i>	<i>String</i>	<i>The customer phone number</i>
<i>contact</i>	<i>String</i>	<i>The customer contact person</i>
<i>fax</i>	<i>String</i>	<i>The customer's fax number</i>
<i>ytdPoints</i>	<i>String</i>	<i>The customer's points total year-to-date</i>
<i>ytdSaved</i>	<i>String</i>	<i>The customer's savings year-to-date</i>

Table 41 <customer> Event

Points Event

points

Sent	When the customer has redeemed or been awarded loyalty points within a transaction.							
XML Format								
<pre><points id="xxxx" idQualifier="xxxx" quantity="1" type="primary" description="Bonus Points" points="1000" redeemed="true false" isVoided="true false"> <pointsTotal type="club1" description="Gold Club Points Total" points="1000" /> <pointsTotal type="club2" description="Baby Club Points Total" points="1000" /> <printLine rawData="[27][5] *** Super Saver Bonus *** " filteredData=" *** Super Saver Bonus *** " /> <printLine rawData=" + 1000 Points " filteredData=" + 1000 Points " /> </points></pre>								
id	String	A unique identifier of the points reward.						
description	String	The description of the points reward.						
redeemed	Boolean	True if the points are redeemed, false if they are awarded.						
pointsTotal	Element	The type, description, and value of the points redeemed or awarded for clubs or secondary accounts.						
isVoided	Boolean	Indicates whether the points have been voided.						
idQualifier	String	Indicates format of the points identifier.						
quantity	String	The number of points						
type	String	The type of points.						
printLine	Element	Includes the raw and filtered print lines that are sent to the POS receipt for this event. <table><tr><td>rawData</td><td>String</td><td>Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as [nn] where nn is the ASCII value of the character.</td></tr><tr><td>filteredData</td><td>String</td><td>The same print line as the rawData, except that any printer control characters have been filtered from the string.</td></tr></table>	rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as [nn] where nn is the ASCII value of the character.	filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.
rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as [nn] where nn is the ASCII value of the character.						
filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.						
points	String	The total points for the primary account.						

Table 42 <points> Event

TransactionTotals Event

transactionTotals

Sent | Whenever an application change occurs which causes an update to any of the transaction totals.

XML Format

<transactionTotals total="xxx.xx" tax="xxx.xx" subTotal="xxx.xx" balanceDue="xxx.xx" foodStampTotal="xxx.xx" foodStampBalance="xxx.xx" foodStampChange="xxx.xx" changeDue="xxx.xx" totalItems="n" totalSavings="xxx.xx" totalCouponCount="n" totalCouponAmount="xxx.xx" foreignBalanceDue="xxx.xx" tenderExchanged="xxx.xx" layawayFee="xx.xx" layawayDeposit="xx.xx" layawayBalanceDue="xxx.xx" />

Note that whenever any of the transaction totals are updated, an event may be dispatched by the application to the framework containing just the values that have been updated.
Values which are not set will return null strings.

total	String	The transaction total amount (including taxes).
tax	String	The transaction tax amount.
subTotal	String	The transaction total (excluding taxes).
balanceDue	String	The remaining tender which is necessary to complete the transaction.
foodStampTotal	String	The total amount of all foodstampable items in the transaction. Should always be included (SA, ACE).
foodStampBalance	String	The remaining transaction balance which may be satisfied via foodstamps. Should be included whenever a foodstamp total is taken. (SA, ACE).
foodStampChange	String	The foodstamp amount due the customer.
changeDue	String	The cash amount due the customer.
totalItems	String	The total number of items in the transaction.
totalSavings	String	The amount saved by the loyalty customer for the transaction (as determined by the POS application).
totalCouponCount	String	The total number of coupons used in this transaction
totalCouponAmount	String	The total value of coupons used in this transaction
foreignBalanceDue	String	The remaining balance translated to the foreign currency as configured in the application. Should be included whenever a foreign total is taken. (SA,ACE, GSA when multiple currency feature is present.)
tenderExchanged	String	The amount of tender accepted in this transaction
layawayFee	String	For layaway transaction type, set to the layaway fee (if any). For layawayCancel transaction type, use a negative layaway fee if the fee is refundable. (GSA)
layawayDeposit	String	For layaway transaction type, set to the layaway deposit amount (if any). For layawayCancel transaction type, use a negative layaway deposit if the deposit is refundable. (GSA) For a layawayPayment transaction type, set to the layaway deposit amount (if any).

layawayBalanceDue	String	For layaway transaction type, set to the transaction amount plus fee amount minus deposit amount. (GSA) For layaway payment transaction type, set to the current layaway balance due (the prior layaway balance due minus payments). (GSA). For layawayCancel transaction type, set to the adjusted layaway balance for item cancels. Set to zero when the entire layaway account is cancelled. In this case, the amountDue should indicate the amount of the refund due the customer. (GSA)
-------------------	--------	--

Table 43 <transactionTotals> Event

TransactionStatus Events

transactionStart		
Sent	Whenever a transaction is started within the POS application. Note: Not sent for entry into training mode. (GSA) Not sent for entry into transaction reentry.	
XML Format		
<pre><transactionStart id="nnn" date="030828" time="15:23" type="cash" modifier="return" category="sales" accountNumber="zzzzzzz" returnPayments="true false" cancelAllItems="true false"/></pre>		
id	String	A unique transaction identifier.
date	String	The transaction's start date (YYMMDD)
time	String	The transaction's start time. (HH:MM)

type	String	<p>The transaction type. Expected values from GSA, SA, & ACE would be:</p> <p>"noSale"</p> <p>"cash" or "regularSale" which are equivalent</p> <p>"cashSpecial"</p> <p>"cashDocument"</p> <p>"cod"</p> <p>"layaway"</p> <p>"chargePlanA"</p> <p>"chargePlanB"</p> <p>"chargePlanC"</p> <p>"chargePlanD"</p> <p>"loan"</p> <p>"pickup" (aka withdrawal)</p> <p>"tenderCount"</p> <p>"totalsReadoutReset"</p> <p>"itemMovementReport"</p> <p>"itemPriceChange"</p> <p>"setTransactionNumber"</p> <p>"offlineReentry"</p> <p>"training"</p> <p>"voidPreviousTransaction"</p> <p>"voidPreviousByLineItem"</p> <p>"tenderListingReport"</p> <p>"tenderRemoval"</p> <p>"terminalMonitor"</p> <p>"queryExchangeRate"</p> <p>"tenderCashing"</p> <p>"tenderExchange"</p> <p>"priceVerify"</p> <p>"terminalTransfer"</p> <p>"terminalProgramLoad"</p> <p>"itemReturn"</p> <p>"wic"</p> <p>"reprintPartialReceipt"</p> <p>"reprintTenderReceipt"</p> <p>"EBTBalanceInquiry"</p> <p>"valueCardBalanceInquiry"</p> <p>"departmentTotalsReport"</p> <p>"layawayPayment"</p> <p>"layawayCancel"</p> <p>"priceChange"</p> <p>"tenderFeeRefund"</p> <p>"suspendedTransactionReport"</p> <p>"modifyDepartmentPresets"</p> <p>"openTransactionReport"</p>
modifier	String	<p>Additional transaction information which may be application specific.</p> <p>"return"</p> <p>"send"</p> <p>"vat"</p> <p>"display"</p>

category	String	Should be either “sales” or “nonsales” depending on the transaction type.
accountNumber	String	Set to the layaway account number when starting a “layaway”, “layawayCancel”, or “layawayPayment” transaction type. (GSA) Set to the charge plan account number when starting a “chargePlanA”, “chargePlanB”, “chargePlanC”, or “chargePlanD” transaction type. (GSA)
returnPayments	Boolean	Included only when the transaction type is “layawayCancel”. Indicates whether payments are returned to the customer (corresponds to customer-initiated). (GSA)
cancelAllItems	Boolean	Included only when the transactiontype is “layawayCancel”. Indicates whether the cancel applies to all items in the layaway account. (GSA)

Table 44 <transactionStart> Event

transactionVoid		
Sent	Whenever a transaction is voided within the POS application. Note: Not sent for exit from training mode (which is a signoff). (GSA) Not sent for exit from transaction reentry (which is a signoff).	
XML Format		
< transactionVoid id="nnn" type="xxx" category="sales nonsales"/>		
id	String	The identifier of the transaction being voided.

type	String	<p>The transaction type. Note: It may not be possible to void all the transaction type listed below. Expected values from GSA, SA, & ACE would be:</p> <p>"noSale"</p> <p>"cash" or "regularSale" which are equivalent</p> <p>"cashSpecial"</p> <p>"cashDocument"</p> <p>"cod"</p> <p>"layaway"</p> <p>"chargePlanA"</p> <p>"chargePlanB"</p> <p>"chargePlanC"</p> <p>"chargePlanD"</p> <p>"loan"</p> <p>"pickup" (aka withdrawal)</p> <p>"tenderCount"</p> <p>"totalsReadoutReset"</p> <p>"itemMovementReport"</p> <p>"itemPriceChange"</p> <p>"setTransactionNumber"</p> <p>"offlineReentry"</p> <p>"training"</p> <p>"voidPreviousTransaction"</p> <p>"voidPreviousByLineItem"</p> <p>"tenderListingReport"</p> <p>"tenderRemoval"</p> <p>"terminalMonitor"</p> <p>"queryExchangeRate"</p> <p>"tenderCashing"</p> <p>"tenderExchange"</p> <p>"priceVerify"</p> <p>"terminalTransfer"</p> <p>"terminalProgramLoad"</p> <p>"itemReturn"</p> <p>"wic"</p> <p>"reprintPartialReceipt"</p> <p>"reprintTenderReceipt"</p> <p>"EBTBalanceInquiry"</p> <p>"valueCardBalanceInquiry"</p> <p>"departmentTotalsReport"</p> <p>"layawayPayment"</p> <p>"layawayCancel"</p> <p>"priceChange"</p> <p>"tenderFeeRefund"</p> <p>"suspendedTransactionReport"</p> <p>"modifyDepartmentPresets"</p> <p>"openTransactionReport"</p>
category	String	Should be either "sales" or "nonsales" depending on the transaction type.

Table 45 <transactionVoid> Event

transactionSuspended		
Sent	Whenever a transaction is suspended within the POS application.	
XML Format		
< transactionSuspended id="nnn" type="xxx" category="sales nonsales"/>		
id	String	The identifier of the transaction being suspended.

type	String	<p>The transaction type. Note: It may not be possible to suspend all the transaction type listed below.</p> <p>Expected values from GSA, SA, & ACE would be:</p> <p>"noSale"</p> <p>"cash" or "regularSale" which are equivalent</p> <p>"cashSpecial"</p> <p>"cashDocument"</p> <p>"cod"</p> <p>"layaway"</p> <p>"chargePlanA"</p> <p>"chargePlanB"</p> <p>"chargePlanC"</p> <p>"chargePlanD"</p> <p>"loan"</p> <p>"pickup" (aka withdrawal)</p> <p>"tenderCount"</p> <p>"totalsReadoutReset"</p> <p>"itemMovementReport"</p> <p>"itemPriceChange"</p> <p>"setTransactionNumber"</p> <p>"offlineReentry"</p> <p>"training"</p> <p>"voidPreviousTransaction"</p> <p>"voidPreviousByLineItem"</p> <p>"tenderListingReport"</p> <p>"tenderRemoval"</p> <p>"terminalMonitor"</p> <p>"queryExchangeRate"</p> <p>"tenderCashing"</p> <p>"tenderExchange"</p> <p>"priceVerify"</p> <p>"terminalTransfer"</p> <p>"terminalProgramLoad"</p> <p>"itemReturn"</p> <p>"wic"</p> <p>"reprintPartialReceipt"</p> <p>"reprintTenderReceipt"</p> <p>"EBTBalanceInquiry"</p> <p>"valueCardBalanceInquiry"</p> <p>"departmentTotalsReport"</p> <p>"layawayPayment"</p> <p>"layawayCancel"</p> <p>"priceChange"</p> <p>"tenderFeeRefund"</p> <p>"suspendedTransactionReport"</p> <p>"modifyDepartmentPresets"</p> <p>"openTransactionReport"</p>
category	String	Should be either "sales" or "nonsales" depending on the transaction type.

Table 46 <transactionSuspended> Event

transactionEnd		
Sent	Whenever a transaction is ended within the POS application.	
XML Format		
< transactionEnd id="nnn" type="xxx" category="sales nonsales"/>		
id	String	The identifier of the transaction being ended.

type	String	<p>The transaction type. Expected values from GSA, SA, & ACE would be:</p> <p>"noSale"</p> <p>"cash" or "regularSale" which are equivalent</p> <p>"cashSpecial"</p> <p>"cashDocument"</p> <p>"cod"</p> <p>"layaway"</p> <p>"chargePlanA"</p> <p>"chargePlanB"</p> <p>"chargePlanC"</p> <p>"chargePlanD"</p> <p>"loan"</p> <p>"pickup" (aka withdrawal)</p> <p>"tenderCount"</p> <p>"totalsReadoutReset"</p> <p>"itemMovementReport"</p> <p>"itemPriceChange"</p> <p>"setTransactionNumber"</p> <p>"offlineReentry"</p> <p>"training"</p> <p>"voidPreviousTransaction"</p> <p>"voidPreviousByLineItem"</p> <p>"tenderListingReport"</p> <p>"tenderRemoval"</p> <p>"terminalMonitor"</p> <p>"queryExchangeRate"</p> <p>"tenderCashing"</p> <p>"tenderExchange"</p> <p>"priceVerify"</p> <p>"terminalTransfer"</p> <p>"terminalProgramLoad"</p> <p>"itemReturn"</p> <p>"wic"</p> <p>"reprintPartialReceipt"</p> <p>"reprintTenderReceipt"</p> <p>"EBTBalanceInquiry"</p> <p>"valueCardBalanceInquiry"</p> <p>"departmentTotalsReport"</p> <p>"layawayPayment"</p> <p>"layawayCancel"</p> <p>"priceChange"</p> <p>"tenderFeeRefund"</p> <p>"suspendedTransactionReport"</p> <p>"modifyDepartmentPresets"</p> <p>"openTransactionReport"</p>
category	String	Should be either "sales" or "nonsales" depending on the transaction type.

Table 47 <transactionEnd> Event

transactionUpdate		
Sent	Whenever required to notify of a significant change in the transaction, depending on the event attributes. ended within the POS application.	
XML Format		
< transactionUpdate <i>property_1</i> ="value_1" <i>property_2</i> ="value_2" ... />		
itemAllowanceAllowed	Boolean	Indicates when the item allowance function is valid in the transaction. (GSA)
itemDiscountAllowed	Boolean	Indicates when the item discount function is valid in the transaction (GSA, ACE)
transactionDiscountAllowed	Boolean	Indicates when the transaction discount function is valid in the transaction. (GSA, SA, ACE)
itemVoidAllowed	Boolean	Indicates when the void line item function is valid in the transaction. (GSA, SA, ACE)

Table 48 <transactionUpdate> Event

transactionTaxChange		
Sent	GSA: When a TAXCODE entry has been applied to the transaction. A <transactionTotals> event should follow this event. ACE & SA: When a transaction discount group has been applied which includes tax exemptions. This event should follow the <transactionDiscount> event, and be followed by a <transactionTotals> event.	
XML Format		
<pre><transactionTaxChange taxType="taxExempt alternateTax manualTax" taxReason="cc" .. isVoided="true false"/></pre>		
taxType	String	(GSA) – Set to the type of taxcode. (SA) – Set to “taxExempt” if the taxReason is a taxExempt discount (ACE) – Set to “taxExempt” if the taxReason is a discount group which exempts all tax plans. Set to “alternateTax” if the taxReason is a discount group which exempts some tax plans.
taxReason	String	(GSA) – The tax code. (SA,ACE) – The discount group.
isVoided	Boolean	Indicates whether the tax change has been voided.

Table 49 <transactionTaxChange> Event

ItemSales Event

itemEvent		
Sent	Whenever the POS application determines that an item has been added to a transaction. Also sent when an item is voided.	
XML Format		
<pre><itemEvent attribute1="xxx" attribute2="xxx" > <printLine rawData="1 Michelob 40oz." " filteredData="1 Michelob 40oz." "/"> <printLine rawData=" 3@3.00 1.00" filteredData=" 3@3.00 1.00"/> <!--This is a yearly restrict period, starting Jan 05 at 12:00AM and ending Feb 25 at 11:59AM → <restrict reversed="false" startMonth="jan" startDate="5" startTime="12:00AM" endMonth="feb" endDate="25" endTime="11:59AM"/> <!--This is a monthly restrict period, starting on the 3rd of each month at 11:00PM, and ending on the 20th of each month at 12:00PM. → <restrict reversed="false" startDate="3" startTime="11:00PM" endDate="20" endTime="12:00PM"/> <!--This is a weekly restrict period, starting on Tuesday at 3:34PM and ending on Saturday at 6:00AM each week. → <restrict reversed="false" startDay="tue" startTime="03:34PM" endDay="sat" endTime="06:00AM"/> <!--This is a daily restrict period, starting at 7:30AM and ending at 8:59PM each day. → <restrict reversed="false" startTime="07:30AM" endTime="08:59PM" /> </itemEvent></pre>		
itemID	String	A unique item identifier. The format of the item-code may be determined by the itemQualifier.
itemIDQualifier	String	Indicates the format of the item code (may be scanned, keyed). For IBM 4690 Supermarket Application, the values may be: ScannedItemCode KeyedItemCode ItemLookupKeyed LinkedItemCode WandedItemCode
description	String	A textual description of the item.
unitPrice	String	The unit price of the item.
quantity	String	The number of items sold for this line item.
weight	String	The weight of this item if sold as a weighted item.
extendedPrice	String	The actual price of the line item.
enteredPriceUsed	String	If a price override or price required item, add enteredPriceUsed="true".
pricingMethod	String	The pricing method. For IBM 4690 Supermarket Application, the values may be: unit dealQuantity basePlus1 groupThreshold reducedDeal increasedDeal alias

dealQuantity	String	The quantity involved in dealQuantity pricing, or other pricing methods which require a quantity.						
dealPrice	String	The price for a deal pricing method. For example, if a deal is 3 for \$1.00, the dealPrice is \$1.00.						
reducedPrice	String	The reduced price may be used by certain pricing methods.						
groupID	String	Identifies the group that the item belongs to for multigroup pricing schemes.						
itemTaxable	Boolean	Indicates whether the item is taxable.						
isVoided	Boolean	Indicates whether the item was voided. If not voided or refunded, then it was sold.						
isReturn	Boolean	Indicates whether the item was returned. If not voided or returned, then it was sold.						
returnReason	String	If the isReturn attribute was set to "true", the field should be included if there is a return reason code. The value should be the return reason code. (ACE)						
isDeposit	Boolean	Indicates whether the item was a deposit. If not voided or refunded, then it was sold.						
foodstampEligible	Boolean	Indicates whether the item may be purchased via foodstamps.						
itemRepeatAllowed	Boolean	Can this item be sold again using a repeat key.						
linkedItemID	String	Item ID linked to this item						
linkedItemIDQualifier	String	Item ID qualifier of item linked to this item						
printLine	Element	<p>Includes the raw and filtered print lines that are sent to the POS receipt for this event.</p> <table> <tr> <td>rawData</td><td>String</td><td>Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as <code>[\nn]</code> where <code>nn</code> is the ASCII value of the character.</td></tr> <tr> <td>filteredData</td><td>String</td><td>The same print line as the rawData, except that any printer control characters have been filtered from the string.</td></tr> </table>	rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as <code>[\nn]</code> where <code>nn</code> is the ASCII value of the character.	filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.
rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as <code>[\nn]</code> where <code>nn</code> is the ASCII value of the character.						
filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.						
requiredAge	String	The number of years old the customer must be to purchase the item (ACE). Note: SA & GSA can support age restricted items only via user extensions. Therefore, part of AEF enabling those applications will require providing this data. I suggest creating a new global string in the AEF CBASIC code, and then documenting that the user exit code should set this string to the required age if the item is age restricted, or an empty string if it is not age restricted.						

wicEligible	Boolean	Indicates whether the item may be purchased via a WIC voucher (SA, ACE).																											
Restrict	Element	<p>Describes any restricted periods for the item. Include a <restrict> element for each time period.</p> <p>(SA,GSA) Include the restrict periods from options for each time restricted item. Note that SA and GSA only allow weekly restricted periods.</p> <p>(ACE) Include the restrict periods from the group identified in the item record. For ACE, restricted periods may be daily, weekly, monthly, and yearly.</p> <p>Attributes:</p> <table> <tr> <td>reversed</td><td>Boolean</td><td>Set to "false" if the item cannot be sold during this <restrict> time period. Set to "true" if this item can only be sold during this <restrict> time period.</td></tr> <tr> <td>startMonth</td><td>String</td><td>The month in which the restrict period begins. Values may be "jan feb mar apr may jun jul aug sep oct nov dec". This attribute is only used for yearly restricted periods (and therefore does not apply to GSA & SA).</td></tr> <tr> <td>endMonth</td><td>String</td><td>The month in which the restrict period ends. Values may be "jan feb mar apr may jun jul aug sep oct nov dec". This attribute is only used for yearly restricted periods (and therefore does not apply to GSA & SA).</td></tr> <tr> <td>startDate</td><td>String</td><td>The numeric day of the month in which the restrict period begins. This attribute is used for yearly and monthly restrict periods (and therefore does not apply to GSA & SA).</td></tr> <tr> <td>endDate</td><td>String</td><td>The numeric day of the month in which the restrict period ends. This attribute is used for yearly and monthly restrict periods (and therefore does not apply to GSA & SA).</td></tr> <tr> <td>startDay</td><td>String</td><td>The day of the week in which the restrict period begins. This attribute is used for weekly restrict periods. Values may be: "sun mon tue wed thu fri"</td></tr> <tr> <td>endDay</td><td>String</td><td>The day of the week in which the restrict period ends. This attribute is used for weekly restrict periods. Values may be: "sun mon tue wed thu fri"</td></tr> <tr> <td>startTime</td><td>String</td><td>The start of the restrict time period in HH:MMAM PM format.</td></tr> <tr> <td>endTime</td><td>String</td><td>The end of the restrict time period in HH:MMAM PM format.</td></tr> </table>	reversed	Boolean	Set to "false" if the item cannot be sold during this <restrict> time period. Set to "true" if this item can only be sold during this <restrict> time period.	startMonth	String	The month in which the restrict period begins. Values may be "jan feb mar apr may jun jul aug sep oct nov dec". This attribute is only used for yearly restricted periods (and therefore does not apply to GSA & SA).	endMonth	String	The month in which the restrict period ends. Values may be "jan feb mar apr may jun jul aug sep oct nov dec". This attribute is only used for yearly restricted periods (and therefore does not apply to GSA & SA).	startDate	String	The numeric day of the month in which the restrict period begins. This attribute is used for yearly and monthly restrict periods (and therefore does not apply to GSA & SA).	endDate	String	The numeric day of the month in which the restrict period ends. This attribute is used for yearly and monthly restrict periods (and therefore does not apply to GSA & SA).	startDay	String	The day of the week in which the restrict period begins. This attribute is used for weekly restrict periods. Values may be: "sun mon tue wed thu fri"	endDay	String	The day of the week in which the restrict period ends. This attribute is used for weekly restrict periods. Values may be: "sun mon tue wed thu fri"	startTime	String	The start of the restrict time period in HH:MMAM PM format.	endTime	String	The end of the restrict time period in HH:MMAM PM format.
reversed	Boolean	Set to "false" if the item cannot be sold during this <restrict> time period. Set to "true" if this item can only be sold during this <restrict> time period.																											
startMonth	String	The month in which the restrict period begins. Values may be "jan feb mar apr may jun jul aug sep oct nov dec". This attribute is only used for yearly restricted periods (and therefore does not apply to GSA & SA).																											
endMonth	String	The month in which the restrict period ends. Values may be "jan feb mar apr may jun jul aug sep oct nov dec". This attribute is only used for yearly restricted periods (and therefore does not apply to GSA & SA).																											
startDate	String	The numeric day of the month in which the restrict period begins. This attribute is used for yearly and monthly restrict periods (and therefore does not apply to GSA & SA).																											
endDate	String	The numeric day of the month in which the restrict period ends. This attribute is used for yearly and monthly restrict periods (and therefore does not apply to GSA & SA).																											
startDay	String	The day of the week in which the restrict period begins. This attribute is used for weekly restrict periods. Values may be: "sun mon tue wed thu fri"																											
endDay	String	The day of the week in which the restrict period ends. This attribute is used for weekly restrict periods. Values may be: "sun mon tue wed thu fri"																											
startTime	String	The start of the restrict time period in HH:MMAM PM format.																											
endTime	String	The end of the restrict time period in HH:MMAM PM format.																											

Table 50 <itemEvent> Event

Coupon Event

coupon

Sent	Whenever the POS application determines that a coupon has been added to a transaction
------	---

XML Format

<Coupon couponType="Store" itemID="80" itemIDQualifier="KeyedItemCode" description="Coupon .25" value="-.25" isVoided="false" isRefunded="false" isDeposit="false" itemRepeatAllowed="false" reducesFoodstampBalanceDue="true" reducesTaxDue="true" pricingMethod="unit" quantity="1" unitPrice=".25" >
<printLine rawData="SCCoupon .25" .25-B" filteredData="SC Coupon .25.25-B"/>
<restrict reversed="false" day="sat" start="12:00AM" end="11:59AM" />
</Coupon>

Note: IBM Electronic Marketing coupons overlay and redefine item record fields to implement various coupon types. These values are shown for completeness, but are implementation details.

itemID	String	A unique identifier for the coupon. The format of the identifier may be determined by the itemIDQualifier.
itemIDQualifier	String	Indicates the format of the identifier (may be scanned, keyed). For IBM 4690 Supermarket Application, the values may be: ScannedItemCode KeyedItemCode ItemLookupKeyed LinkedItemCode WandedItemCode
description	String	A textual description of the coupon.
couponType	String	"Store", "Manufacturer", etc.
manufacturerNumber	String	The coupon manufacturer number. (Valid only for Manufacturer coupons)
value	String	The actual value of the coupon.
reducesFoodstampBalanceDue	Boolean	Indicates whether the coupon reduces the foodstamp balance due.
reducesTaxDue	Boolean	Indicates whether this coupon reduces the tax balance due.
itemTaxable	Boolean	Indicates whether the coupon is taxable.
isVoided	Boolean	Indicates whether the coupon was voided. If not voided, then it was applied to the transaction.
isRefunded	Boolean	Indicates whether the coupon was refunded. If not voided or refunded, then it was sold.
requiredAge	String	The number of years old the customer must be to redeem the coupon.
wicEligible	Boolean	Is this coupon eligible for WIC

printLine	Element	Includes the raw and filtered print lines that are sent to the POS receipt for this event.		
		rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as [nn] where nn is the ASCII value of the character.
		filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.
restrict	Element	Describes any restricted periods for the coupon. Attributes: reversed="true false" day="sat sun mon tue wed thu fri" start="HH:MMA M PM" end="HH:MM AM PM"		
itemModifier	String	Optionally gives more information about the type of coupon. This is POS application specific.		
itemRepeatAllowed	String	Can this coupon be applied again using a repeat key.		
The following attributes are item record fields used by the 4690 EM implementation of coupons:				
unitPrice	String	The unit value of the coupon.		
pricingMethod	String	The pricing method. For IBM 4690 Supermarket Application, the values may be: unit dealQuantity basePlus1 groupThreshold reducedDeal increasedDeal alias		
quantity	String	The number of times that the coupon was applied.		
weight	String	The weight of this item if sold as a weighted item.		
dealQuantity	String	The quantity involved in dealQuantity pricing, or other pricing methods which require a quantity.		
dealPrice	String	The price for a deal pricing method. For example, if a deal is 3 for \$1.00, the dealPrice is \$1.00.		
reducedPrice	String	The reduced price may be used by certain pricing methods.		
isDeposit	Boolean	Indicates whether the coupon was a deposit. If not voided or refunded, then it was sold.		

<i>groupID</i>	<i>String</i>	<i>Identifies the group that the item belongs to for multigroup pricing schemes.</i>
----------------	---------------	--

Table 51 <coupon> Event

Discount Events

transactionDiscount		
Sent	Whenever the POS application applies or voids a discount to a transaction.	
XML Format		
<pre><transactionDiscount description="AARP discount" isVoided="true false" discountMethod="percent allowance" rate="5.0" amount="0.50" discountReason="xx" reducesTaxBalanceDue="true false" > <printLine rawData=" AARP discount -0.50" filteredData=" AARP discount -0.50"/> </printLine> </transactionDiscount ></pre>		
description	String	A textual description of the discount. The attribute should be provided if the application is able to supply the description.
isVoided	Boolean	Indicates whether the discount is being added or voided.
discountMethod	String	Either “percent” for a percentage off or “allowance” for a fixed amount off.
rate	String	The percentage off for a “percent” method discount.
amount	String	The actual amount of the discount. For SA & ACE, include this field only if the discount rate is non-zero.
reducesTaxBalanceDue	Boolean	Indicates whether the discount reduces the tax balance due. For SA & ACE, include this field only if the discount rate is non-zero.
discountReason	String	The discount reason code or discount group. (GSA, SA, ACE).

printLine	Element	Includes the raw and filtered print lines that are sent to the POS receipt for this event.		
		rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as [\nn] where nn is the ASCII value of the character.
		filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.

Table 52 <transactionDiscount> Event

Note: In the case of a tax exemption, the discountReason can be used to look up the transactionDiscountReason in POSDataProvider to find out the tax exemption information for the discount.

lineItemDiscount		
Sent	Whenever the POS application applies or voids a line item discount.	
XML Format		
<pre><lineItemDiscount description="AARP discount" isVoided="true false" appliesTo="next" discountMethod="percent allowance" rate="5.0" amount="0.50" discountType="01" discountReason="xxxxxx" reducesTaxBalanceDue="true false"> <printLine rawData=" AARP discount -0.50" filteredData=" AARP discount -0.50"/> </lineItemDiscount></pre>		
description	String	A textual description of the discount.
isVoided	Boolean	Indicates whether the discount is being added or voided.
appliesTo	String	Indicates the line item the discount applies to. Values may be "previous", "next" or the number of the line item within the transaction (assuming the first line item is item #1).
discountMethod	String	Either "percent" for a percentage off or "allowance" for a fixed amount off.
rate	String	The percentage off for a "percent" method discount.
reducesTaxBalanceDue	Boolean	Indicates whether the discount reduces the tax balance due.
amount	String	The actual amount of the discount.

printLine	Element	Includes the raw and filtered print lines that are sent to the POS receipt for this event.		
		rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as <code>[\nn]</code> where <code>nn</code> is the ASCII value of the character.
		filteredData	String	The same print line as the <code>rawData</code> , except that any printer control characters have been filtered from the string.
discountReason	String	The discount reason code or discount group. (GSA, ACE).		
Optional data not included in 4690 implementation				
discountType	String	Identifies the type of discount. This may be application specific information.		

Table 53 <lineItemDiscount> Event

Tender Event

Tender

Sent | Whenever the POS application determines that a credit tender has been added or voided.

XML Format

```
<tender tenderType="xxx" tenderVariety="yyy" isVoided="true|false" description="VISA"
amount="35.23" currency="USD" fee="0.50" accountNumber="3893828382" referenceNumber="121665">
  <printLine rawData="VISA #####888      35.23"
              filteredData="VISA #####888      35.23" />
  <printLine rawData="      3893828382      "
              filteredData="      3893828382      " />
</tender>
```

tenderType	String	The type of tender (as defined by the POS application).
tenderVariety	String	The variety of tender (as defined by the POS application). Note that the concatenation of the tenderType and tenderVariety is used as a lookup key in tendermap.properties to determine the Java event to construct from the <tender> message.
isVoided	String	A flag that tells us if the tender was voided.
description	String	A textual description of the tender.
amount	String	The amount of the tender.
currency	String	The currency of the tender as defined by the three-character country code in ISO 4217. See www.bsi-global.com/iso4217currency . Note: the IBM 4690 POS applications do not provide this data.
fee	String	Any fee charged for accepting the tender.
accountNumber	String	The account number of the tender.
expirationDate	String	Used for credit tenders. MMY format.
cardID	String	Optionally used for credit tenders when the entry of the tender requires an additional card ID to be input. This is usually an extension of the account number printed on the back of the card.
referenceNumber	String	The reference number of the tender.

printLine	Element	Includes the raw and filtered print lines that are sent to the POS receipt for this event.		
		rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as <code>[\nn]</code> where <i>nn</i> is the ASCII value of the character.
		filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.

Table 54 <tender> Event

CashReceipt Event

cashReceipt		
Sent	Whenever the POS application writes to the printer	
XML Format		
<pre><cashReceipt linefeeds="2" paperCut="false"> <printLine rawData="[\27][\05]VISA #####888 35.23" filteredData="VISA #####888 35.23"/> <printLine rawData=" approval: 1123 " filteredData=" approval: 1123 "/> </cashReceipt></pre>		
linefeeds	String	Indicates the number of lines advanced
paperCut	Boolean	Indicates if a receipt paper cut was performed
rawData	String	Contains the print data including any printer control characters. Any characters that are not representable in an XML document (such as those with ASCII values below 32) are transformed as [nn] where nn is the ASCII value of the character.
filteredData	String	The same print line as the rawData, except that any printer control characters have been filtered from the string.

Table 55 <cashReceipt> Event

Scale Event

scale		
Sent	Whenever the POS application reads the scale for a weight item.	
XML Format		
<pre><scale scaleWeightValue="2.67" scaleWeightUnit="LB" scaleWeightLabels="ACE SurePOS" /></pre>		
scaleWeightValue	String	Weight of the item as a formatted string
scaleWeightUnit	String	Weight unit of measure
scaleWeightLabels	String	Optional data. Used for displaying weights and measures information.

Table 56 <scale> Event

Report Event

report	
Sent	Whenever the POS application has report information available. (Note: ACE uses this event for OTR.)
XML Format	

<pre> <report id="OTR"> <section id="1"> <line type="0" data="TERMINAL DATE TIME AMOUNT TRANS." /> </section> <section id="2"> <line type="0" data="000 7900 9093 9022." /> <line type="0" data="001 8809 9093 9022." /> </section> </report> </pre>				
id	String	Report identifier		
section	element	id	String	Identifies a section of the report or a function to be performed.
line	element	type	String	Identifies the type of line or attributes of the line to be displayed.
		data	String	The data to be displayed.

Table 57 <report> Event

DataEvent

dataEvent

Sent	To set one or more POSDataProvider property values. Provides generic extension mechanism for setting properties with the AEF POSDataProvider.	
XML Format		
<dataEvent dataCategory="xxxxxx" PROPERTY_NAME1="yyyyyy" PROPERTY_NAME2="ZZZZ" />		
dataCategory	String	Set to one of the one of the CATEGORY constant values in one of the xxxProperties.java classes in the com.ibm.AEF.data package. May also be a user category.
PROPERTY_NAME1	String	PROPERTY_NAME1 is the name of the property which will have its value set.

Table 58 <dataEvent> Event

The following table provides a list of dataEvents currently provided. Data events generate AEFPropertyChangeEvents through the AEF POSDataProvider interface using the dataCategory and PROPERTY_NAME values specified in the xml.

Description	AEF XML Message
Options are being loaded into POS terminal sales application, or have completed loading	<dataEvent dataCategory="WORKSTATION_STATUS" OPTIONS_LOADING_IN_PROGRESS="true false"/>
The substate of the terminal sales application has changed. Substates provide a more granular state change mechanism.	<dataEvent dataCategory="POS_DEVICE" subState="xxxx" />
A POS transaction unit of work has started or ended. A unit of work defines a logical beginning and end	<dataEvent dataCategory="WORKSTATION_STATUS" unitOfWork="true false" />

of a task within a transaction (e.g., add an item, applying a discount, etc).	
The OPERATOR_DISPLAY property informs the AEF which logical display device is used for prompting the operator (cashier) at the POS terminal.	<dataEvent dataCategory="POS_DEVICE" OPERATOR_DISPLAY="LineDisplay1 LineDisplay2 LineDisplay3"/>
The CUSTOMER_DISPLAY property informs the AEF which logical display device is used for prompting the customer at the POS terminal.	<dataEvent dataCategory="POS_DEVICE" CUSTOMER_DISPLAY=" LineDisplay1 LineDisplay2 LineDisplay3"/>
The terminal is being used in training mode.	<dataEvent dataCategory="WORKSTATION_STATUS" TRAINING_MODE="true false"/>
The terminal has been disabled (GSA)	<dataEvent dataCategory="WORKSTATION_STATUS" terminalDisabled="true false"/>
The terminal is being used in transaction reentry mode (GSA).	<dataEvent dataCategory="WORKSTATION_STATUS" REENTRY_MODE="true false"/>

Table 59 <dataEvent> Values

WorkstationStatus Event

WorkstationStatus Event

workstationStatus		
Sent	To notify listeners that a change has occurred in the POS workstation (terminal). The terminalStatus property specifies the current state of the terminal. Additional workstation <i>properties</i> may also be specified if needed. WorkstationStatus events are typically used for events that are transitory (such as a “clear” or “acknowledge”), where a property value does not need to be maintained or persisted.	
XML Format		
<workstationStatus terminalStatus="xxxx" additionalProperty1="xxx" additionalProperty2="xxx" />		
terminalStatus	String	The current status of the terminal device.
additionalPropertyN	String	Specify any other data needed for this event

Table 60 <workstationStatus> Event

OptionsEvent

Options	
Sent	Whenever the POS terminal sales application loads or modifies options or configuration data which specify the behavior of the POS application.
XML Format	

```

<options>
  <store name="Raleigh Store #1201" number="1201" phone="919-555-1212" />

  <!-- Price override reason codes (0 to N allowed) -->
  <priceOverrideReason description="desc1" code="code1"/>
  .
  .
  <priceOverrideReason description="descN" code="codeN"/>

  <!-- Refund reason codes (0 to N allowed) -->
  <refundReason description="desc1" code="code1"/>
  .
  .
  <refundReason description="descN" code="codeN"/>

  <!-- Void reason codes (0 to N allowed) -->
  <voidReason description="desc1" code="code1"/>
  .
  .
  <voidReason description="descN" code="codeN"/>

  <!-- Tare codes (0 to N allowed) -->
  <tareCode description="desc1" code="code1"/>
  .
  .
  .
  <tareCode description="descN" code="codeN"/>

  <!-- Transaction discount reason codes (0 to N allowed) -->
  <transactionDiscountReason description="xxx" code="nnn" rate="zz.z"
    taxExemption="true|false"
    taxPlan1Exempt="true|false"
    taxPlan2Exempt="true|false"
    taxPlan3Exempt="true|false"
    taxPlan4Exempt="true|false"
    taxPlan5Exempt="true|false"
    taxPlan6Exempt="true|false"
    taxPlan7Exempt="true|false"
    taxPlan8Exempt="true|false"/>
  .
  .
  <!-- Item discount reason codes (0 to N allowed) -->
  <itemDiscountReason description="xxx" code="nnn" rate="zz.z"/>
  .
  .
  <!--Tender definitions (0 to N allowed) -->
  <tenderDefinition description="CASH" shortDescription="CSH" type="01" variety="01" key="xxx"
    foreignTender="true|false" allowed="true|false" canBeLoaned="true|false"
    canBePickedUp="true|false" canBeCounted="true|false" canBeRefunded="true|false"
    canBeVerified="true|false" tenderExchangeRank="05" />
  .
  .
  <!--VAT Tax Codes (0 to N allowed) -->
  <vatTaxCode code="xx" percent="nn.n"/>
  .
  .

```

```

<storeOptions alphaStateInputAllowed="true|false"
alphaDriversLicenseInputAllowed="true|false"
priceVerifyInsideTransactionAllowed="true|false"
multipleCashDrawerSupport="true|false"
  overrideRequiresManagerKey="true|false"
maximumTransactionSize="nn"
transactionWarningSize="nn"
  foodstampsAllowed="true|false"
onlyFoodstampsAllowedAfterFoodstampTotal="true|false"
maximumSuspendedTransactions="nn"
customerFunctionCode="nn"
  suspendTransactionAllowed="true|false"
  WICTenderOnlyInWICTransaction="true|false"
  weightInputDecimalPlaces="x"
volumeInputDecimalPlaces="3"
volumeUnitPriceDecimalPlaces="3"
enableGiftReceiptPrinting="true|false"
  foreignTenderAllowed="true|false"
  openTransReportEnabled="true|false"
  openTransPrintEnabled="true|false"
  openTransRefreshEnabled="true|false"
  openTransFunctionsEnabled="true|false"
  couponMultiplierEnabled="true|false"
  scanManagerID="true|false"
  eatInTakeoutPromptEnabled="true|false">
  <transactionDefinition transactionType="noSale"
    promptForAccountNumber="true|false"
    documentInsertUsed="true|false"
    paymentsAllowed="true|false"
    allowancesAllowed="true|false"
    discountsAllowed="true|false"
    returnsAllowed="true|false"
    promptForOriginalSalesperson="true|false"
    promptForTermsOfSale="true|false"
    voidTransactionAllowed="true|false"
    suspendTransactionAllowed="true|false" />
  <transactionDefinition transactionType="regularSale"
    promptForAccountNumber="true|false"
    documentInsertUsed="true|false"
    paymentsAllowed="true|false"
    allowancesAllowed="true|false"
    discountsAllowed="true|false"
    returnsAllowed="true|false"
    promptForOriginalSalesperson="true|false"
    promptForTermsOfSale="true|false"
    voidTransactionAllowed="true|false"
    suspendTransactionAllowed="true|false" />
  <transactionDefinition transactionType="cashSpecial" ... />
  <transactionDefinition transactionType="cashDocument" ... />
  <transactionDefinition transactionType="cod" ... />
  <transactionDefinition transactionType="layaway" ... />
  <transactionDefinition transactionType="chargePlanA" ... />
  <transactionDefinition transactionType="chargePlanB" ... />
  <transactionDefinition transactionType="chargePlanC" ... />
  <transactionDefinition transactionType="chargePlanD" />

</storeOptions>

```

<pre> <terminalOptions noSaleTransactionAllowed="true false" cashTransactionAllowed="true false" layawayTransactionAllowed="true false" itemReturnTransactionAllowed="true false" voidTransactionAllowed="true false" taxCodeRequired="true false" operatorIDRequired="true false" cashSpecialTransactionAllowed="true false" cashDocumentTransactionAllowed="true false" chargePlanATransactionAllowed="true false" chargePlanBTransactionAllowed="true false" chargePlanCTransactionAllowed="true false" chargePlanDTransactionAllowed="true false" codTransactionAllowed="true false" sendTransactionAllowed="true false" paymentsAllowed="true false" allowanceAllowed="true false" discountsAllowed="true false" originalSalesPersonRequired="true false" termsOfSaleRequired="true false" suspendTransactionAllowed="true false" ...> </terminalOptions> <!-- Department definitions (0 to N allowed) --> <departmentDefinition description="aaaaaa" number="xxxx" key="nn" toggleTaxAllowed="true false managerOverride operatorOverride" toggleFoodstampAllowed="true false managerOverride operatorOverride" refundAllowed="true false managerOverride operatorOverride" storeCouponAllowed="true false managerOverride operatorOverride" manufacturerCouponAllowed="true false managerOverride operatorOverride" priceAllowed="true false managerOverride operatorOverride" /> . . <!-- Alternate Tax codes (0 to N allowed) --> <alternateTaxCode code="xx" description="yyyyyy"/> . . <!-- Manual Tax codes (0 to N allowed) --> <manualTaxCode code="xx" description="yyyyyy"/> . . <!-- No Tax codes (0 to N allowed) --> <noTaxCode code="xx" description="yyyyyy"/> . . </options> </pre>																																
store	Element	<p>General information about the store (include those attributes which the application has data for)</p> <table> <tr><td>name</td><td>String</td><td>Store name</td></tr> <tr><td>number</td><td>String</td><td>Store number</td></tr> <tr><td>division</td><td>String</td><td>Division</td></tr> <tr><td>address1</td><td>String</td><td>Address line 1</td></tr> <tr><td>address2</td><td>String</td><td>Address line 2</td></tr> <tr><td>city</td><td>String</td><td>City</td></tr> <tr><td>state</td><td>String</td><td>State</td></tr> <tr><td>zip</td><td>String</td><td>Zipcode</td></tr> <tr><td>phone1</td><td>String</td><td>Phone Number 1</td></tr> <tr><td>phone2</td><td>String</td><td>Phone Number 2</td></tr> </table>	name	String	Store name	number	String	Store number	division	String	Division	address1	String	Address line 1	address2	String	Address line 2	city	String	City	state	String	State	zip	String	Zipcode	phone1	String	Phone Number 1	phone2	String	Phone Number 2
name	String	Store name																														
number	String	Store number																														
division	String	Division																														
address1	String	Address line 1																														
address2	String	Address line 2																														
city	String	City																														
state	String	State																														
zip	String	Zipcode																														
phone1	String	Phone Number 1																														
phone2	String	Phone Number 2																														

priceOverrideReason	Element	(ACE). Include a separate <refundReason> element for each price override reason code. Each <refundReason> includes the following attributes:		
		Description	String	Optional textual description of the discount reason. For example, "employee discount". Include if the application is able to provide the description.
		Code	String	The price override reason code.
refundReason	Element	(ACE). Include a separate <refundReason> element for each refund reason code. Each <refundReason> includes the following attributes:		
		description	String	Optional textual description of the refund reason. For example, "spoiled produce". Include if the application is able to provide the description.
		code	String	The refund reason code.
tareCode	Element	(SA, ACE). Includes a separate <tareCode> element for each tare code. Each <tareCode> includes the following attributes:		
		description	String	Optional textual description of the tare reason. For example, "medium meat tray". Include if the application is able to provide the description.
		code	String	The numeric tare code.

transactionDiscountReason	Element	(GSA, SA, ACE). Include a separate <transactionDiscountReason> element for each transaction discount reason or group. Each <transactionDiscountReason> includes the following attributes:		
		description	String	Optional textual description of the discount reason. For example, "medium meat tray". Include if the application is able to provide the description.
		code	String	The numeric discount reason. For SA & ACE, this is the discount group.
		rate	String	The discount percentage (for example, "10.0" for a ten percent discount. (SA, ACE).
		taxExemption	Boolean	(SA) Set to true if the discount group is a tax exemption. (ACE) Set to true if the discount group exempts all active tax plans.
		taxPlan1Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 1.
		taxPlan2Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 2.
		taxPlan3Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 3.
		taxPlan4Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 4.
		taxPlan5Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 5.
		taxPlan6Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 6.
		taxPlan7Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 7.
		taxPlan8Exempt	Boolean	(ACE) Set to true if the discount group is set to exempt tax plan 8.

itemDiscountReason	Element	(GSA, ACE). Include a separate <itemDiscountReason> element for each item discount reason or group. In ACE, only discount groups which do not include tax exemptions may be used as item discounts. Therefore, no discount groups which include tax exemptions should be included as an item discount reason. Each <itemDiscountReason> includes the following attributes:		
		description	String	Optional textual description of the discount reason. For example, "medium meat tray". Include if the application is able to provide the description.
		code	String	The numeric discount reason. For ACE, this is the discount group.
		rate	String	The discount percentage (for example, "10.0" for a ten percent discount. (ACE).

tenderDefinition	Element	(GSA, SA, ACE). Include a separate <tenderDefinition> element for each tender in the application. Each <tenderDefinition> includes the following attributes:		
		description	String	Textual description of the tender. For example, "CASH. (GSA, SA, ACE)
		shortDescription	String	Provided only if the app has short descriptions for the tender. In GSA, this is the "Tender Foreign Currency 2 (or 3) currency identifier" in store options for the tender. For GSA, only foreign tenders will have a shortDescription. For ACE, this is the "Short Description" in the tender definition. SA does not support short descriptions for tenders.
		type	String	The tender type as defined by the application. For example, "4". (GSA, SA, ACE)
		variety	String	The tender variety as defined by the application. For example, "2". (SA, ACE)
		key	String	The function code of the tender on the keyboard. (GSA)
		foreignTender	Boolean	Indicates if the tender is a foreign tender. (GSA, SA, ACE)
		allowed	Boolean	Indicates if the register accepts this tender. (GSA, SA, ACE)
		canBeLoaned	Boolean	(SA) Set true. (ACE) Set according to application definition.
		canBePickedUp	Boolean	(SA) Set true. (ACE) Set according to application definition.
		canBeCounted	Boolean	(SA) Set true. (ACE) Set according to application definition.
		canBeRefunded	Boolean	(SA) Set to true. (ACE) Set according to application definition.
		canBeVerified	Boolean	(SA) Set to true. (ACE) Set according to application definition.
		tenderExchangeRate	String	(SA,ACE) Set according to application definition.
vatTaxCode	Element	(GSA). Include a separate <vatTaxCode> element for each VAT tax code in the application. Each <vatTaxCode> includes the following attributes:		
		code	String	The numeric tax code.
		Percent	String	The tax percentage (for example, "10.0" for a ten percent tax.

storeOptions	Element	The <storeOptions> element includes the following attributes:		
		alphaStateInputAllowed	Boolean	Indicates whether alphabetic input is allowed for state input. (ACE)
		alphaDriversLicenseInputAllowed	Boolean	Indicates whether alphabetic input is allowed for drivers license input. (ACE)
		priceVerifyInsideTransactionAllowed	Boolean	Indicates whether the operator may perform a price verification procedure inside a sales transaction. (SA, ACE)
		overrideRequiresManagerKey	Boolean	Indicates if the manager key is required to complete a manager override. (SA, ACE)
		maximumTransactionSize	String	The maximum number of items the application allows within a single transaction. (SA, ACE)
		transactionWarningSize	String	The number of items at which a transaction size warning is generated. (SA, ACE)
		foodstampsAllowed	Boolean	Indicates whether foodstamps are accepted in the store. (SA, ACE)
		onlyFoodstampsAllowedAfterFoodstampTotal	Boolean	Indicates whether foodstamps must be used after a foodstamp total. (SA, ACE)
		maximumSuspendedTransactions	String	The maximum number of transactions allowed to be suspended per terminal or operator. (SA, ACE)
		customerFunctionCode	String	The keyboard function code used for customer loyalty number entry. (SA-EM, ACE)
		suspendTransactionAllowed	Boolean	Indicates whether suspending transactions is allowed in the store. (SA, ACE)
		WICTenderOnlyInWICTransaction	Boolean	Indicates whether WIC is the only tender allowed in a WIC transaction. (SA, ACE)
		weightInputDecimalPlaces	String	The number of significant digits after the decimal for keyboard weight input. (SA, ACE)
		openTransReportEnabled	Boolean	(ACE)
		openTransPrintEnabled	Boolean	(ACE)
		openTransRefreshEnabled	Boolean	(ACE)
		openTransFunctionsEnabled	Boolean	(ACE)
		couponMultiplierEnabled	Boolean	(ACE)
		eatInTakeoutPromptEnabled	Boolean	(ACE)
		scanManagerID	Boolean	(ACE)

		foreignTenderAllowed	Boolean	Indicates whether foreign tender is allowed in the store. (SA, ACE)
		transactionDefinition	Element	(GSA) For each of the 9 transaction types, include a <transactionDefinition> element which lists the transactionType and attributes for the 10 functions indicating whether they are allowed or disallowed.
		volumeInputDecimalPlaces	Boolean	The number of significant digits for volume input (ACE)
		volumeUnitPriceDecimalPlaces	Boolean	The number of significant digits for pricing by volume (ACE)
		enableGiftReceiptPrinting	Boolean	Enable the printing of gift receipts. (ACE)
<transactionDefinition>	Element	(GSA) The <transactionDefinition> element is included for each of the following transaction types: noSale, regularSale, cashSpecial, cashDocument, cod, layaway, chargePlanA, chargePlanB, chargePlanC, chargePlanD. It includes the following attributes:		
		transactionType	String	One of the 9 values above.
		promptForAccountNumber	Boolean	As set in GSA options.
		documentInsertUsed	Boolean	As set in GSA options.
		paymentsAllowed	Boolean	As set in GSA options.
		allowancesAllowed	Boolean	As set in GSA options.
		discountsAllowed	Boolean	As set in GSA options.
		returnsAllowed	Boolean	As set in GSA options.
		promptForOriginalSalesperson	Boolean	As set in GSA options.
		promptForTermsOfSale	Boolean	As set in GSA options.
		voidTransactionAllowed	Boolean	As set in GSA options.
		suspendTransactionAllowed	Boolean	As set in GSA options.

<terminalOptions>	Element	The <terminalOptions> element includes the following attributes:		
		noSaleTransactionAllowed	Boolean	(GSA)
		cashTransactionAllowed	Boolean	(GSA).
		layawayTransactionAllowed	Boolean	(GSA)
		itemReturnTransactionAllowed	Boolean	(GSA)
		voidTransactionAllowed	Boolean	(GSA)
		taxCodeRequired	Boolean	(GSA)
		operatorIDRequired	Boolean	(GSA)
		cashSpecialTransactionAllowed	Boolean	(GSA)
		cashDocumentTransactionAllowed	Boolean	(GSA)
		chargePlanATransactionAllowed	Boolean	(GSA)
		chargePlanBTransactionAllowed	Boolean	(GSA)
		chargePlanCTransactionAllowed	Boolean	(GSA)
		chargePlanDTransactionAllowed	Boolean	(GSA)
		codTransactionAllowed	Boolean	(GSA)
		sendTransactionAllowed	Boolean	(GSA)
		paymentsAllowed	Boolean	(GSA)
		allowanceAllowed	Boolean	(GSA)
		discountsAllowed	Boolean	(GSA)
		originalSalesPersonRequired	Boolean	(GSA)
		termsOfSaleRequired	Boolean	(GSA)
		suspendTransactionAllowed	Boolean	(GSA)

departmentDefinition	Element	Include a separate <departmentDefinition> element for each department defined in the application. Each <departmentDefinition> includes the following attributes:		
		description	String	A textual description of the department, for example, "Meat". Include this attribute if the application is able to provide the data.
		number	String	The department number.
		key	String	The function code on the keyboard which is used to ring sales for this department.
		toggleTaxAllowed	String	(SA) Values are "true" or "false". (ACE) Values are "true", "false", "managerOverride", or "operatorOverride".
		toggleFoodstampAllowed	String	(SA) Values are "true" or "false". (ACE) Values are "true", "false", "managerOverride", or "operatorOverride".
		refundAllowed	String	(SA) Values are "true" or "false". (ACE) Values are "true", "false", "managerOverride", or "operatorOverride".
		storeCouponAllowed	String	(SA) Values are "true" or "false". (ACE) Values are "true", "false", "managerOverride", or "operatorOverride".
		manufacturerCouponAllowed	String	(SA) Values are "true" or "false". (ACE) Values are "true", "false", "managerOverride", or "operatorOverride".
		priceAllowed	String	(ACE) Values are "true", "false", "managerOverride", or "operatorOverride".

alternateTaxCode	Element	<p>(GSA) Include a separate <alternateTaxCode> element for each alternate tax code defined in the application.</p> <p>(ACE) Include a separate <alternateTaxCode> for each discount group with a zero discount rate, and which does not exempt all the tax plans.</p> <p>Each <alternateTaxCode> element includes the following attributes:</p>		
		description	String	A textual description of the tax code or discount group, for example, "ILL state tax only". Include this attribute if the application is able to provide the data.
		code	String	(GSA) The tax code. (ACE) The discount group.
manualTaxCode	Element	<p>(GSA) Include a separate <manualTaxCode> element for each manual tax code defined in the application.</p> <p>Each <manualTaCodex> element includes the following attributes:</p>		
		description	String	A textual description of the tax code. Include this attribute if the application is able to provide the data.
		code	String	The tax code.
noTaxCode	Element	<p>(GSA) Include a separate <noTaxCode> element for each zero tax code defined in the application.</p> <p>(SA) Include a separate <noTaxCode> element for each discount group with a discount rate of zero, and taxExemption set true.</p> <p>(ACE) Include a separate <noTaxCode> element for each discount group with a discount rate of zero, and tax exemptions of all tax plans.</p> <p>Each <noTaxCode> element includes the following attributes:</p>		
		description	String	A textual description of the tax code or discount group. Include this attribute if the application is able to provide the data.
		code	String	(GSA) The tax code. (SA,ACE) The discount group.

Table 61 <options> Event

Appendix E. SALogonActionImpl Source Example

This section explains how “action” classes work by walking through a particular class: `SALogonActionImpl.java`. This class is used to implement the logon function when any of the `POSAutomationProvider.logon` methods are called.

The `POSAutomationProvider.logon` method was either passed, or constructed an `OperatorIdentifier` and placed in in the argument list of the `ActionRequest`. The `OperatorIdentifier` includes the operator ID and password. On line 64-66, the operator ID and passwords are retrieved and stored in member variables. If no operator id or password was provided on the logon call, defaults are retrieved from the “aplogon” bundle on lines 70 and 74. See “Default Operator ID/Passwords” on page 140.

The first significant line of code in the `performAction` method is the call to the `super.performAction` method on line 105. The superclasses check for errors and allow the AEF Error Handler to attempt to clear them before proceeding with the action.

On line 129-132, the values of several SA states are retrieved into member variables.

On lines 144 and 151 a state check is done to see if SA is in the “secure signoff” state, or the “signon” state. If neither, an exception is thrown because SA is not in a proper state to perform the signon.

Assuming that SA is in the “signon” state, line 168 calls a method to send the operator ID key sequence. If the operator ID key sequence is sent successfully, line 173 calls a method to send the operator password key sequence. If the operator password key sequence is sent successfully, line 178 calls a method to wait for the AEF to detect and create an `Operator` object to be returned.

Line 215 is the start of the `sendIDSequence` method. The `SimpleKeySequenceActionImpl` for sending the `<operator-id><SLASH>` key sequence is setup on lines 236-241. “args” is a `HashMap` which is used to contain the arguments for the key sequence action. The `HashMap` is cleared on line 236. On line 238, the sequence id is set. The sequence id is used by the factory to lookup the corresponding key sequence in the “sequence” bundle (see “Key Sequence Bundle” on page 260). The key sequences defined in the bundle contain variables of the form `%n`. On line 237, the variable “%0” is given the value of the operator id.

The “good” conditions for this key sequence is defined on line 242. In this case, the single good condition is that SA ends up in the “password” state after sending the `<operator-id><SLASH>` key sequence. If there are multiple conditions defined in the condition array, they are logically or’d.

A new `ConditionLock` instance is created on line 248. This lock will be used to block the calling thread while the key sequence is sent to the POS application and to wait for a good condition, bad condition, or a timeout to occur.

On line 253, the current instance number of the last `Operator` detected by the `ObjectDetector` is retrieved and stored. This is crucial to knowing when a new `Operator` instance has been created. This instance number will be passed to an `ObjectDetectorLock` on line 435 in order to wait for a new instance of the `Operator` to be detected.

The next step is to have the ConditionLock perform the SimpleKeySequenceActionImpl to send the <operator-id><SLASH> key sequence as shown on line 254. The good conditions are passed, along with the standard set of bad conditions from [SABadConditionsImpl.java](#). For SA, a common back condition is that the application goes into state 1 (clear state). The performActionAndWait method will block until one of the good or bad conditions is observed, or a timeout occurs. The integer returned from the performActionAndWait method indicates which condition was observed. A zero return value indicates the first good condition was observed. A "one" return value indicates the second good condition was observed, etc. A "negative one" return value indicates the first bad condition was observed. A "negative two" indicates the second bad condition was observed, etc. If a negative return value is observed, an AEFErrorHandler is created to attempt to deal with the error on lines 263 and 264.

Once the operator ID sequence has been sent successfully, the sendPasswordSequence method is called. This method which begins on line 297. This method is similar to the sendIDSequence method.

Once the operator password sequence has been sent successfully, the waitForOperator method is called. The waitForOperator method begins on line 420, and uses an ObjectDetectorLock to block the calling thread until a new Operator instance is detected, or until a timeout occurs. The Operator instance number which was saved earlier is passed to the ObjectDetectorLock on line 435 to ensure that only a new Operator is detected.

If a new Operator is successfully detected, the performAction return value is set to reference the new Operator instance.

```

1.  /*
2.   * SALogonActionImpl
3.   *
4.   * 07/29/2002
5.   *
6.   * Copyright:
7.   * Licensed Materials - Property of IBM
8.   * "Restricted Materials of IBM"
9.   * 5724-AEF
10.  * (C) Copyright IBM Corp. 2003.
11.  *
12.  * %W% %E%
13.  */
14. package com.ibm.retail.AEF.action;
15.
16. import com.ibm.retail.AEF.automation.*;
17. import com.ibm.retail.AEF.util.*;
18. import com.ibm.retail.si.util.*;
19. import com.ibm.retail.si.Copyright;
20. import com.ibm.retail.AEF.factory.*;
21. import com.ibm.retail.AEF.thread.*;
22. import com.ibm.retail.AEF.data.*;
23. import com.ibm.retail.AEF.session.*;
24.
25. import java.util.*;
26. import java.rmi.*;
27.
28. import org.apache.commons.logging.Log;
29. import org.apache.commons.logging.LogFactory;
30.
31. /**
32.  * SALogonActionImpl is a class which the POSAutomationProvider uses to accomplish
33.  * a logon onto the SA application.
34.  */

```

```

35. */
36. public class SALogonActionImpl extends SAActionImpl
37. {
38.
39.     static String copyright()
40.     { return com.ibm.retail.si.Copyright.IBM_COPYRIGHT_SHORT; }
41.     private static AEFPerfTrace perfTrace = AEFPerfTrace.getInstance();
42.
43.     /**
44.      * Constructor
45.      *
46.      * @param request The ActionRequest which contains a HashMap of arguments.
47.      *
48.      * @exception AEFException
49.      */
50.     public SALogonActionImpl(ActionRequest request) throws AEFException
51.     {
52.         super(request);
53.
54.         if (log.isTraceEnabled())
55.         {
56.             tempAEFMessage.setMessage("+Enter ALogonActionImpl.SALogonActionImpl().");
57.             log.trace(tempAEFMessage);
58.         }
59.
60.         if (sessionID == null)
61.         {
62.             sessionID = "999";
63.         }
64.         OperatorIdentifier opId = (OperatorIdentifier) (request.getArguments());
65.         id = opId.getID();
66.         password = opId.getPassword();
67.         newPassword = opId.getNewPassword();
68.         if (id == null || id.length() == 0)
69.         {
70.             id = AppLogons.getID(sessionID);
71.         }
72.         if (password == null || password.length() == 0)
73.         {
74.             password = AppLogons.getPassword(sessionID);
75.         }
76.
77.         if (log.isTraceEnabled())
78.         {
79.             tempAEFMessage.setMessage("-Exit SLogonActionImpl.SALogonActionImpl().");
80.             log.trace(tempAEFMessage);
81.         }
82.     }
83.
84.     /**
85.      * Perform the action represented by the ActionRequest and return an ActionResult.
86.      *
87.      *
88.      *
89.      * @param request The ActionRequest which contains the classname and arguments.
90.      * @return Object The Operator instance.
91.      * @exception AEFException
92.      *         Among the possible error codes are:
93.      *         <br>AEFConst.INVALID_ARGUMENT, AEFConst.NEW_PASSWORD_PROHIBITED
94.      *         <br>AEFConst.APPLICATION_NOT_IN_PROPER_STATE
95.      */
96.     public Object performAction() throws AEFException
97.     {
98.         if (log.isTraceEnabled())
99.         {
100.             tempAEFMessage.setMessage("+Enter SLogonActionImpl.
101.                                     performAction().");
102.             log.trace(tempAEFMessage);
103.         }
104.
105.         super.performAction(); // Call super to perform any common
106.                                // processing and clear any errors

```

```

107.                                     // before we start.
108.
109.     if (newPassword != null && newPassword.length() > 0)
110.     {
111.         String tempString =
112.             "SALogonActionImpl.performAction(): Change password
113.             function not implemented.";
114.         AEFException tempException = new AEFException(
115.             AEFConst.INVALID_ARGUMENT,
116.             AEFConst.NEW_PASSWORD_PROHIBITED,
117.             tempString);
118.         tempAEFMessage.setMessage(tempString);
119.         log.error(tempAEFMessage, tempException);
120.         throw tempException;
121.     }
122.
123.     Object retVal = null;
124.     int seqResult = -1;
125.     String currentState = getCurrentState();
126.     detector = ((DetectorAccess)(SessionContext.getSession())).
127.                 getOperatorDetector();
128.
129.     idState = State.getState("ID");
130.     passwordState = State.getState("PASSWORD");
131.     itemEntryState = State.getState("ITEMENTRY");
132.     securedSignoffState = State.getState("SECURED");
133.
134.     // Check if we are in a clear state.
135.     // This could be a B014 NO TRANSACTION RECOVERY
136.     // for example. If so, clear the error.
137.     if (currentState.equals(State.getState("CLEAR")))
138.     {
139.         sendClearSequence(true);
140.         currentState = getCurrentState();
141.     }
142.
143.
144.     if (currentState.equals(securedSignoffState))
145.     {
146.         // An operator is already signed on.
147.         retVal = doSecuredSignon();
148.     }
149.     //Check precondition that POS_STATE is "ID" state (2).
150.     //If not, throw exception.
151.     else if (!currentState.equals(idState))
152.     {
153.         // Not in valid state for signon.
154.
155.         String tempString = "SALogonActionImpl.performAction(): POS
application not in proper state to perform logon.\n Expected state " + idState + ",
but application is in state " + currentState + ".";
156.         AEFException tempException = new AEFException(
157.             AEFConst.APPLICATION_NOT_IN_PROPER_STATE,
158.             0,
159.             tempString);
160.         tempAEFMessage.setMessage(tempString);
161.         log.error(tempAEFMessage, tempException);
162.         throw tempException;
163.     }
164.     else
165.     {
166.         // Application is in correct state to start performing the logon.
167.
168.         seqResult = sendIDSequence();
169.         if (seqResult==0)
170.         {
171.             // Successfully got to password state,
172.             // send the password sequence.
173.             seqResult = sendPasswordSequence();
174.             if (seqResult == 0 ||
175.                 seqResult == 1)
176.             {

```

```

177.         Operator operator = null;
178.         operator = waitForOperator();
179.         retVal = operator;
180.         try
181.         {
182.             operator.getInfo().setPassword(password);
183.         }
184.         catch (RemoteException re)
185.         {
186.             // Can't get here.
187.
188.             tempAEFMessage.setMessage("There was a remote exception
thrown in SALogonActionImpl.performAction().");
189.             log.error(tempAEFMessage, re);
190.         }
191.     }
192. }
193.
194.
195.     if (log.isTraceEnabled())
196.     {
197.         tempAEFMessage.setMessage("-Exit
SALogonActionImpl.performAction().");
198.         log.trace(tempAEFMessage);
199.     }
200.     return retVal;
201. }
202.
203.
204.
205. /**
206.  * Sends the key sequence for entering the operator id.
207.  *
208.  *
209.  * @return int 0 = success,
210.  * @exception AEFException
211.  *     Among the possible error codes are:
212.  *     <br>AEFConst.CONFIG_ERROR, AEFConst.NO_DEFAULT_ID_CONFIGURED
213.  */
214.
215. public int sendIDSequence() throws AEFException
216. {
217.     if (log.isTraceEnabled())
218.     {
219.         tempAEFMessage.setMessage("+Enter
SALogonActionImpl.sendIDSequence().");
220.         log.trace(tempAEFMessage);
221.     }
222.     int retVal = -1;
223.
224.
225.     if ( id == null || id.length() == 0)
226.     {
227.         // Error, unable to lookup default id for terminal number.
228.
229.         String tempString = "SALogonActionImpl: Default ID and/or password
not configured for terminal " + sessionId + ", unable to perform logon.\nSee
applogon.properties for id/password configuration.";
230.         AEFException tempException = new AEFException(AEFConst.CONFIG_ERROR,
AEFConst.NO_DEFAULT_ID_CONFIGURED, tempString);
231.         tempAEFMessage.setMessage(tempString);
232.         log.error(tempAEFMessage, tempException);
233.         throw tempException;
234.     }
235.
236.     args.clear();
237.     args.put("%0", id);
238.     args.put("SEQUENCE_ID", "id");
239.     keySequenceAction = (AEFAction)
240.         (actionFactory.makeAction(new
241.             ActionRequest("SimpleKeySequenceAction", args)));
242.     Condition[] goodConditions =

```

```

243.         {
244.             new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
245.                                         POSDeviceProperties.POS_STATE,
246.                                         passwordState),          // 0 = success
247.         };
248.         lock = new ConditionLock();
249.
250.         // Save any current operator instance number so we
251.         // are sure to only get an operator
252.         // created after this key sequence is sent.
253.         instanceNumber = detector.getInstanceNumber();
254.         retVal = lock.performActionAndWait("wait-for-password-state",
255.                                           keySequenceAction,
256.                                           goodConditions,
257.                                           BadConditionsImpl.
258.                                               getInstance().getBadConditions(),
259.                                               getTimeout());
260.
261.         if (retVal < 0)
262.         {
263.             AEFErrorHandler errorHandler = new AEFErrorHandler("Send Signon ID
Sequence Error");
264.             retVal = errorHandler.handleError(lock,
265.                                               keySequenceAction,
266.                                               goodConditions,
267.                                               BadConditionsImpl.
268.                                                   getInstance().
269.                                                       getBadConditions(),
270.                                                       getTimeout(),
271.                                                       retVal);
272.         }
273.         if (log.isDebugEnabled())
274.         {
275.             tempAEFMessage.setMessage("SALogonActionImpl.sendIDSequence()
returning : " + retVal);
276.             log.debug(tempAEFMessage);
277.         }
278.         if (log.isTraceEnabled())
279.         {
280.             tempAEFMessage.setMessage("-Exit
SALogonActionImpl.sendIDSequence().");
281.             log.trace(tempAEFMessage);
282.         }
283.         return retVal;
284.     }
285.
286.
287.
288.     /**
289.      * Sends the key sequence for entering the operator password.
290.      *
291.      *
292.      * @return int 0,1 = success
293.      * @exception AEFException
294.      *      Among the possible error codes are:
295.      *      <br>AEFConst.CONFIG_ERROR, AEFConst.NO_DEFAULT_ID_CONFIGURED
296.      */
297.     public int sendPasswordSequence() throws AEFException
298.     {
299.         if (log.isTraceEnabled())
300.         {
301.             tempAEFMessage.setMessage("+Enter
SALogonActionImpl.sendPasswordSequence().");
302.             log.trace(tempAEFMessage);
303.         }
304.
305.         int retVal = -1;
306.
307.         if ( password == null || password.length() == 0)
308.         {
309.             // Error, unable to lookup default password for terminal number.
310.             AEFException ex = new AEFException(AEFConst.CONFIG_ERROR,

```

```

311.                                     AEFConst.NO_DEFAULT_ID_CONFIGURED,
312.                                     "SALogonActionImpl: Default ID
and/or password not configured for terminal " + sessionID + ", unable to perform
logon.\nSee applogon.properties for id/password configuration.");
313.                                     try
314.                                     {
315.                                         sendClearSequence();
316.                                     }
317.                                     catch (AEFException e)
318.                                     {
319.                                         // Ignore any errors from sending the clear.
320.                                     }
321.                                     tempAEFMessage.setMessage("SALogonActionImpl: Default ID and/or
password not configured for terminal " + sessionID + ", unable to perform logon.\nSee
applogon.properties for id/password configuration.");
322.                                     log.error(tempAEFMessage, ex);
323.                                     throw ex;
324.                                 }
325.
326.
327.                                 // We have a password which we can use to attempt the
328.                                 // secure mode signon.
329.                                 args.clear();
330.                                 args.put("%0", password);
331.                                 args.put("SEQUENCE_ID", "password");
332.                                 keySequenceAction = (AEFAction)
333.                                     (actionFactory.makeAction(new ActionRequest("SimpleKeySequenceAction",
334.                                                                                     args)));
335.
336.                                 Condition andConds1[] =
337.                                 {
338.                                     new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
339.                                                                 POSDeviceProperties.POS_STATE,
340.                                                                 State.getState("MAIN")),
341.                                     new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
342.                                                                 POSDeviceProperties.POS_SUB_STATE,
343.                                                                 Substate.getSubstate("SELECT_PROCEDURE"))
344.                                 };
345.
346.
347.                                 Condition andConds2[] =
348.                                 {
349.                                     new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
350.                                                                 POSDeviceProperties.POS_STATE,
351.                                                                 State.getState("MAIN")),
352.                                     new PropertyEqualsCondition(POSDeviceProperties.CATEGORY,
353.                                                                 POSDeviceProperties.POS_SUB_STATE,
354.                                                                 Substate.getSubstate("CHANGE_OR_BAL_DUE"))
355.                                 };
356.
357.
358.                                 Condition goodConditions[] =
359.                                 {
360.                                     new AndCondition(andConds1),
361.                                     new AndCondition(andConds2)
362.                                 };
363.
364.                                 if (lock == null)
365.                                 {
366.                                     lock = new ConditionLock();
367.                                 }
368.                                 if (log.isDebugEnabled())
369.                                 {
370.                                     tempAEFMessage.setMessage("About to wait on the perform action for
'password'");
371.                                     log.debug(tempAEFMessage);
372.                                 }
373.                                 if (perfTrace.isEnabled(AEFPerfTrace.COARSE))
374.                                 {
375.                                     perfTrace.reportTimer(AEFPerfTrace.COARSE,
376.                                                             sessionID,

```

```

377.                                     "logon",
378.                                     ">>>Sending logon password key sequence to
application.");
379.     }
380.     retVal= lock.performActionAndWait("wait-for-item-entry-state",
381.                                     keySequenceAction,
382.                                     goodConditions,
383.                                     BadConditionsImpl.
384.                                     getInstance().getBadConditions(),
385.                                     getTimeout());
386.     if (retVal < 0)
387.     {
388.         AEFErrorHandler errorHandler = new AEFErrorHandler("Send Signon
Password Sequence Error");
389.         retVal = errorHandler.handleError(lock,
390.                                     keySequenceAction,
391.                                     goodConditions,
392.                                     BadConditionsImpl.
393.                                     getInstance().
394.                                     getBadConditions(),
395.                                     getTimeout(),
396.                                     retVal);
397.     }
398.     if (log.isDebugEnabled())
399.     {
400.         tempAEFMessage.setMessage(" Condition returned was " + retVal);
401.         log.debug(tempAEFMessage);
402.     }
403.     if (log.isTraceEnabled())
404.     {
405.         tempAEFMessage.setMessage("-Exit
SALogonActionImpl.sendPasswordSequence().");
406.         log.trace(tempAEFMessage);
407.     }
408.     return retVal;
409. }
410.
411.
412.
413.
414. /**
415.  * Blocks the calling thread until a new operator object is created.
416.  *
417.  *
418.  * @exception AEFException
419.  */
420. public Operator waitForOperator() throws AEFException
421. {
422.     if (log.isTraceEnabled())
423.     {
424.         tempAEFMessage.setMessage("+Enter
SALogonActionImpl.waitForOperator().");
425.         log.trace(tempAEFMessage);
426.     }
427.
428.     Operator retVal = null;
429.     // Successfully got to item entry. Wait for the operator
430.     // object to be created.
431.     ObjectDetectorLock objLock = new ObjectDetectorLock();
432.     retVal = (Operator)
433.         (objLock.waitForNewObject("wait-for-operator",
434.                                 detector,
435.                                 instanceNumber,
436.                                 getTimeout()));
437.     if (perfTrace.isEnabled(AEFPerfTrace. COARSE))
438.     {
439.         perfTrace.reportTimer(AEFPerfTrace. COARSE,
440.                               sessionID,
441.                               "logon",
442.                               "<<<Detected operator logon from application.");
443.     }
444.

```

```

445.         if (log.isTraceEnabled())
446.         {
447.             tempAEFMessage.setMessage("-Exit
SALogonActionImpl.waitForOperator().");
448.             log.trace(tempAEFMessage);
449.         }
450.         return retVal;
451.     }
452.
453.
454.
455.     /**
456.      * Performs the logic required to sign an operator on from secured mode.
457.      *
458.      *
459.      * @exception AEFException
460.      *      Among the possible error codes are:
461.      *      <br>AEFConst.PROCEDURE_NOT_ALLOWED,
462.      *      AEFConst.ANOTHER_OPERATOR_SIGNED_ON
463.      */
464.     public Operator doSecuredSignon() throws AEFException
465.     {
466.         if (log.isTraceEnabled())
467.         {
468.             tempAEFMessage.setMessage("+Enter
SALogonActionImpl.doSecuredSignon().");
469.             log.trace(tempAEFMessage);
470.         }
471.         if (log.isDebugEnabled())
472.         {
473.             tempAEFMessage.setMessage("App in secured signoff state.");
474.             log.debug("App in secured signoff state.");
475.         }
476.         Operator operator = null;
477.         // Somebody is already signed on, check to see
478.         // if the id of the new signon request
479.         // matches the id of the already signed on operator.
480.
481.         try
482.         {
483.             operator = automationProvider.getOperator();
484.             if (operator != null && !operator.getInfo().getID().equals(id))
485.             {
486.                 // Currently signed on operator is a different operator.
487.
488.                 String tempString = "SALogonActionImpl.performAction(): Can't
perform logon for operator " + id + " because operator " + operator.getInfo().getID()
+ " is already signed on.";
489.                 AEFException tempException = new
490.                     AEFException(AEFConst.PROCEDURE_NOT_ALLOWED,
491.                         AEFConst.ANOTHER_OPERATOR_SIGNED_ON,
492.                         tempString);
493.                 tempAEFMessage.setMessage(tempString);
494.                 log.error(tempAEFMessage, tempException);
495.                 throw tempException;
496.             }
497.             else if (operator != null)
498.             {
499.                 // The requested operator is already
500.                 // signed on, but app is in secured
501.                 // signoff mode, just need to send password sequence.
502.
503.                 // Use the password of the currently
504.                 // signed on operator if possible.
505.                 String pwd = operator.getInfo().getPassword();
506.                 if (pwd != null && pwd.length() > 0)
507.                 {
508.                     password = pwd;
509.                 }
510.                 int seqResult = sendPasswordSequence();
511.                 if (seqResult == 0 ||
512.                     seqResult == 1)

```

```

513.         {
514.             operator = waitForOperator();
515.             try
516.             {
517.                 operator.getInfo().setPassword(password);
518.             }
519.             catch (RemoteException re)
520.             {
521.                 // Can't get here.
522.
523.                 tempAEFMessage.setMessage("There was a remote exception
thrown in SALogonActionImpl.doSecuredSignon().");
524.                 log.error(tempAEFMessage, re);
525.             }
526.         }
527.     }
528. }
529. catch (RemoteException re)
530. {
531.     // Not expected to get here since we are calling locally.
532.
533.     tempAEFMessage.setMessage("There was a remote exception thrown in
SALogonActionImpl.doSecuredSignon().");
534.     log.error(tempAEFMessage, re);
535. }
536.
537. if (log.isTraceEnabled())
538. {
539.     tempAEFMessage.setMessage("-Exit
SALogonActionImpl.doSecuredSignon().");
540.     log.trace(tempAEFMessage);
541. }
542. return operator;
543. }
544.
545. /* Instance Variables */
546. protected AEFAction          keySequenceAction;
547. protected String             id;
548. protected String             password;
549. protected String             newPassword;
550. protected ConditionLock      lock;
551. protected ObjectDetector     detector;
552. protected int                instanceNumber;
553. protected String             idState;
554. protected String             passwordState;
555. protected String             itemEntryState;
556. protected String             securedSignoffState;
557.
558. private static Log log = LogFactory.getLog(SALogonActionImpl.class);
559. }

```

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering the subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility

or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Trademarks

The following are trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM
the IBM logo

Everyplace
WebSphere

Microsoft, Windows, Windows NT, Windows 2000 and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, or other countries, or both.

IceElite is a trademark or registered trademark of Icesoft Technologies, Inc. in the United States, other countries, or both.

This product includes software developed by the MX4J project.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

Other company, product, and service names may be trademarks or service marks of others.