WebSphere™ Studio Device Developer

User’s Guide for Service Management Framework Toolkit

Version 3.1

Note!

Before using this information and the product it supports, read the information in “Notices” on page 49.

1st Edition (September, 2002)

This product includes software developed by the Eclipse Project (`http://www.eclipse.org/`).

© Copyright International Business Machines Corporation 1999, 2002. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1999, 2002

Chapter 1. Concepts

Service Management Framework (SMF) is an implementation of the Open Services Gateway Initiative (OSGi™) organization's Service Platform Release 2 (SPR2) specification. The OSGi organization is an industry group that defines and promotes an open standard for the networked delivery of managed services to local networks and devices. An open standard, meaning a standard that is created and voluntarily adhered to by manufacturers, has the advantage of minimizing the number of products that are incompatible and therefore exclusive of other standards. The Open Services Gateway standard is intended to complement other residential standards and is open to almost any protocol, transport or device layer, and is therefore inclusive rather than exclusive of other standards.

OSGi specification contents

The OSGi™ SPR2 specification defines a Framework on which applications can run. Developers can write new applications and adapt existing applications to run on the Framework. The Framework acts as a layer that allows operators to deploy multiple applications on a single Java™ Virtual Machine (JVM). Application developers partition applications into services and other resources. Services and resources are packaged into bundles, which are files that serve as the delivery unit for applications. Bundles have manifests with special headers that enable the sharing of classes and services at the package level.

The OSGi Specification also defines a set of services for applications to use, which run within the framework. These services include:

- Configuration Admin Service - allows an operator to set the configuration information of deployed bundles.
- Device Access - supports automatic detection of attached and detached hardware devices and can automatically download and start appropriate device drivers.
- HTTP Service - provides users with access to services on the Internet and other networks.
- Log Service - provides a general purpose message logger for the OSGi environment.
- Package Admin Service - allows a management bundle to provide the policies for package sharing.
- Permission Admin Service - allows a management bundle to administer bundle's permissions and provides defaults for all bundles.
- Preferences Service - provides a persistent data store for bundles.
- User Admin Service - provides a lightweight authentication function.

Service Management Framework

IBM's Service Management Framework (SMF) is a production-ready software management framework for network-delivered applications, to better meet the needs of internet ready device manufacturers and service providers, such as telcos, ISPs, cable companies, and power utilities.

SMF is the core technology. It is packaged within two products, the SMF Toolkit, and SMF Bundle Developer. The SMF Toolkit is available as a free download from http://www-3.ibm.com/pvc/products/wes_embedded.

© Copyright IBM Corp. 1999, 2002

This document discusses only the SMF Toolkit.

SMF execution environment

SMF runs on any environment that supports Java 2 (for example, WSDD 5.0).

OSGi™ Specification concepts

This section contains information on the following concepts defined by the OSGi™ Specification:

- Framework
- Bundles
- Manifest files
- Services
- Packages
- Bundle Activator

Framework

The framework is the core of the OSGi™ implementation. It manages the installation and update of bundles in an OSGi environment, and manages the dependencies between bundles and services.

Bundles

A bundle is the smallest unit of management for the Framework. It is a Java Archive (JAR) file or a J9 Executable (JXE) with a manifest that contains special headers. These headers describe the bundle to SMF and state the bundle's dependencies. A bundle can register services with SMF that can be used by other bundles.

Like ordinary JAR files, JAR bundle files typically contain java class files and resources. JXE bundle files do not contain individual java class files. Instead the class files are pre-linked into a single file named `rom.classes`. JXE files can be stored and executed in ROM, which minimizes application startup times and RAM requirements.

Where bundles differ from ordinary JAR files is in the descriptive information found in the manifest file. Non-bundle JAR files often keep very little information in the manifest file. A bundle's manifest will usually contain descriptive information such as the bundle's name and version. It will also frequently identify packages and services the bundle requires or makes available to other bundles.

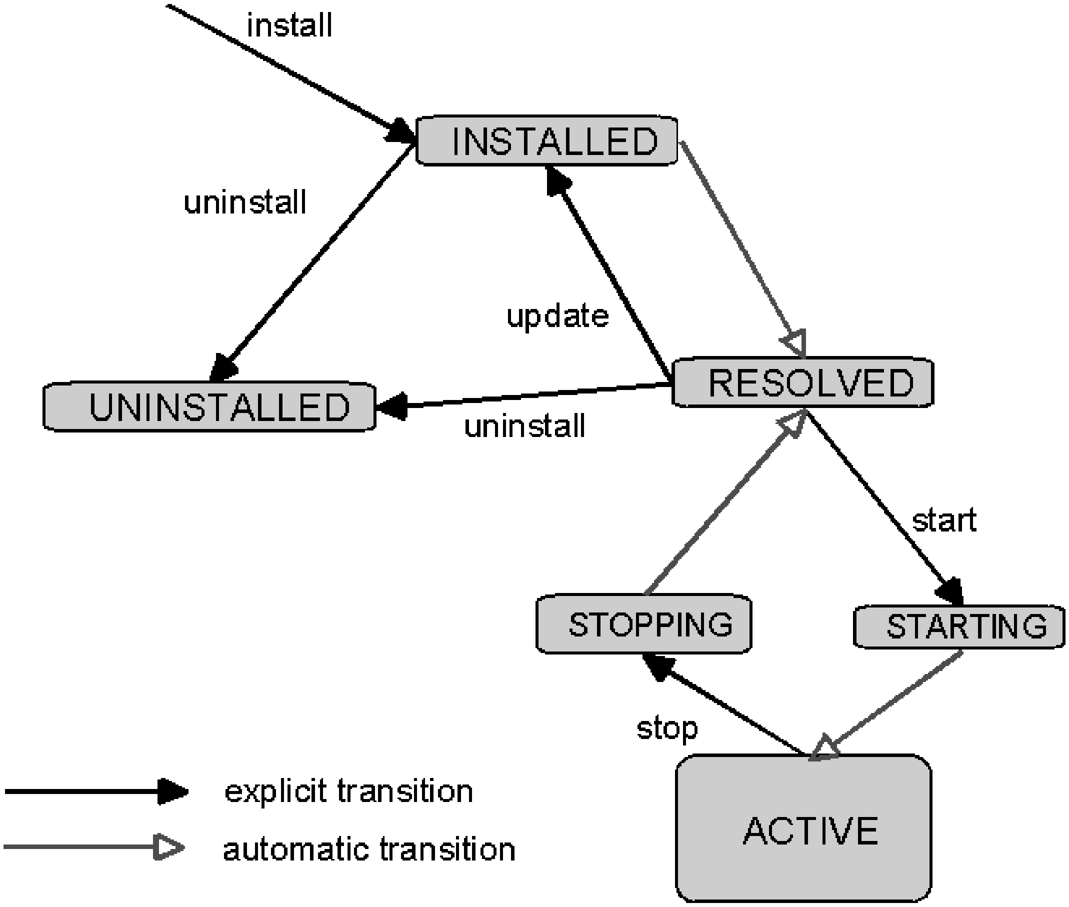
Bundle life cycle

The framework manages the life cycle of bundles. As you install and run a bundle, it goes through various states. The possible states of a bundle are:

- INSTALLED - the bundle has been installed, but all it's dependencies have not been met. It requires packages that have not yet been exported by any currently installed bundle.
- RESOLVED - the bundle is installed, and it's dependencies have been met, but it is not running. This state might be skipped when a bundle is started if all dependencies are met.
- STARTING - a temporary state while a bundle is starting.
- ACTIVE - the bundle is running.

- STOPPING - a temporary state while a bundle is stopping.
- UNINSTALLED - the bundle no longer exists in the framework.

The following diagram demonstrates the relationship between the SMF console commands and the bundle's states.



Manifest files

Bundles must contain a manifest file. The bundle's manifest file contains data that the framework needs to correctly install and activate the bundle. The OSGi™ Specification defines a set of headers that you must use to define this data. You must place the OSGi headers at the start of the manifest file. Any headers that are not recognized are ignored by SMF. Refer to the OSGi SPR2 specification for more information about the OSGi Manifest file format and syntax.

Manifest files must reside in the META-INF directory inside the bundle and must be named MANIFEST.MF Case sensitivity is important. For example, the MANIFEST.MF file will not be found if it is in the meta-inf directory instead of the META-INF directory.

Services

A service is defined by a Java interface. Any class that implements this interface is a provider of the service. Services are used to decouple bundles that provide services from bundles that make use of the services.

The only code a service provider and a service user must share is the service definition itself.

A bundle that makes use of a service that it does not provide can make this fact known to the framework by including an import-service header in its manifest. However, this is not required. A service is actually imported into a bundle at runtime when code within that bundle requests a provider of the service from the framework.

A bundle that provides services can optionally include an export-service header in its manifest. A service is exported by a bundle at runtime when code within that bundle makes a provider available to the framework.

The import-service and export-service tags inform the SMF Bundle Server of a bundle's prerequisites. While these tags are optional, if the tags are not defined, the installation of prerequisite bundles will not work.

Packages

Bundles can use code defined elsewhere by importing packages. Though it is possible to construct a bundle that does not rely on any classes outside the Java base packages, most bundles import code from other bundles or the base runtime class path.

Any class used within a bundle that is neither defined in the bundle nor is a base Java class (that is, those within packages beginning with java.) must be imported into the bundle. To accomplish this, include an import clause for the class's package in the bundle's manifest. You can only explicitly import whole packages; individual classes cannot be explicitly imported.

A bundle can make classes it defines available to other bundles by exporting packages. To allow other bundles to access a particular package, an export clause for the package is included in the manifest of the bundle containing the package.

Bundle Activators

A bundle can include a class that implements the *org.osgi.framework.BundleActivator* interface. This class is identified to the framework by a Bundle-Activator header in the bundle's manifest. At startup time, the framework creates an instance of this class and calls its `start()` method. The activator can then publish services, start its own threads, and so on.

At shutdown, the activator's `stop()` method is called. The activator can take this opportunity to release resources obtained since `start()` was received as well as revoke published services.

Prerequisites

Developing bundles using SMF requires that you have the following:

- Runtime environnment that supports Java 2 (for example WebSphere Studio Device Developer version 5.0)
- Runtime environnment that supports Java 2 (for example WebSphere Studio Device Developer version 5.0), including a Java compiler
- Understanding of Java programming and servlets

The default development environment for SMF is Microsoft Windows 2000 and WebSphere Studio Device Developer 5.0.

Chapter 2. Getting Started with SMF Toolkit

This chapter contains information that will help you get started with the SMF Toolkit.

Throughout this document SMFHOME refers to the directory where you installed SMF.

Starting the SMF Toolkit from the command line

This section describes how to start the SMF Toolkit from the command line.

smf.properties

You can specify properties for SMF in the directory where you installed SMF. If you want to use a different properties file, start SMF with the Java command option `-Dcom.ibm.osg.smf.properties=`*`properties file`* , where *properties file* represents the path to the properties file you want to use. You can find descriptions of other properties used by SMF in the `smf.properties` file.

The smf.properties file adds the properties contained within it to the system properties. The values in the smf.properties file can be overridden at launch time with the Java -D launch option.

SMF launching options

To launch the SMF Toolkit, use one of the batch files, as appropriate for your environment that reside in the directory where you installed SMF.

Refer to the following sections for more information:

- Batch files for starting the SMF Toolkit
- Java command line arguments
- Security
- SMF Launcher command line arguments

Batch files for starting SMF Toolkit

Table 1. Batch files for starting SMF Toolkit

Batch file	Platform	Description
smf.bat	Windows	Windows® command that starts SMF using the WebSphere® Studio Device Developer 5.0 toolkit.
smfjdk.bat	Windows	Windows command that starts SMF using the JDK toolkit.
smf	Linux	Shell command that starts SMF using the WebSphere Studio Device Developer 5.0 toolkit.
smfjdk	Linux	Shell command that starts SMF using the JDK toolkit.

The batch file invokes the `com.ibm.osg.smf.SMFLauncher` class. You can configure launch options for SMF using command-line arguments, or by modifying the smf.properties file. Command line arguments override the smf.properties file contents. The smf.properties file resides in the directory where you installed SMF.

© Copyright IBM Corp. 1999, 2002

Java command line arguments

Table 2. Java command line arguments

Argument	Description
-Djava.security.manager	Sets the value of the java system property java.security.manager to indicate that the default security manager will be invoked. Both this argument and the java.security.policy argument must be used to properly enable security within SMF.
-Djava.security.policy=smf.policy	Sets the value of the java system property java.security.policy to indicate that the security policy smf.policy should be used by the security manager. Both this argument and the java.security.manager argument must be used to properly enable security within SMF.
-Dcom.ibm.osg.smf.bundledir	Defines the location where SMF stores installed bundles and their data. SMF is configured to use either of the following two directories: jxebundles |jarbundles You can also specify a custom directory.

Security

If you choose to start SMF with a SecurityManager (for example, with the Java command options `-Djava.security.manager -Djava.security.policy=smf.policy`), then SMF registers the Permission Admin service, provides bundles with their configured permissions, and checks OSGi™ permissions as specified. If you choose not to start SMF with Security Manager, then SMF will not register the Permission Admin service and will not check OSGi permissions.

The policy file, `smf.policy`, provides `smf.jar` with AllPermission.

See the Simple permission policy documentation for an example of setting bundle permissions using the Permission Admin service. All SMF bundles include a `permission.info` file describing the bundle's requested permissions.

SMF Launcher command line arguments

Argument	Description
-con[sole][:port]	Starts SMF with a console window. Any command line arguments not recognized are passed to the console for it to execute. If a port is specified the console will listen on that port for commands. If no port is specified, the console will use System.in and System.out.
-r[esourceManagement]	Enables resource management. If this option is not specified, resource management is not enabled.

Argument	Description
-platform[:platform-name][platform-args]	`[platform-name] := "" | fully qualified class name [platform-args] := *( ":" [value]) [value] := [token] | [quoted-string] For example: 1) -platform::"bundledir=c:\jarbundles":reset   DefaultPlatform is chosen with args[]= {"bundledir=c:\jarbundles", "reset"} 2) -platform:com.foo.MyPlatform` There are two platform classes: • com..ibm..osg.smf.platform.DefaultPlatform. Supports a file-based file system. • com.ibm.osg.smf.platform.flash.FlashBundleStore Supports a flash file system. -platform specifies the implementation class for the platform to be used. args contains a list of platform arguments, separated by ":". Platform arguments format is defined by the platform implementation class. They are passed to the platform class as an array of Strings. The DefaultPlatform class recognizes and handles the following options: • **bundledir=***directory name* The directory to be used by thte platform to store data. If a directory name is specified, the platform initializes to store bundles in that directory. This overrides values set in either the smf.properties file or with a -D launch option. If this argument contains a blank space or the ":" character, for example, "bundledir=c:\my dir", then you must enclose this argument in single or double quotation marks (" " or ' '). • **reset** Resets Perform the reset action to clear the bundledir. Any other arguments are ignored. The FlashBundleStore class recognizes and handles the following options: • **filename=***flashSimulatorFileName*For running with the Flash Simulator on development platforms. If fileName is not specified, then the file name is taken from the FLASH_FILE environment variable and if that variable is not defined, then the default name *flash_file.dat* is used. values set in either the smf.properties file or with a -D launch option. If this argument contains a blank space or the ":" character, for example, "filename=my file", then you must enclose this argument in single or double quotation marks (" " or ' '). • **filesize=***size* The size in MB (default 4) in which the flash simulator file will be created if it does not exist or the existing file with flashSimulatorFileName has a different size. • **reset**Clears the contents of the bundle store. (Flash simulator only.) • **compact**Compacts the bundle store. (flash simulator only.) If **-platform** is not specified, or if no platform classname is specified, DefaultPlatform will be used, which is file based and stores the files in the \bundles directory relative to the current directory.

Any other command line arguments are passed on to the console window of the framework if started with the `-console` option. If the console is not started, any unrecognized arguments will be ignored and a message displayed.

Note: You can only issue console commands when running the toolkit on the target device with the `-console` option.

If you do not specify any launch options, SMF is started with the following options:

- with the Default Platform
- without a console window

- without resource management
- without the remote agent

SMF Console commands

You can control the function of SMF from within the SMF Console. This section describes the following:

- Command argument definitions
- SMF-defined commands
- Memory space-related commands

Command argument definitions

The following table describes the arguments that you can use with SMF Console commands:

Command arguments	description
<id>	The bundle ID. The unique number assigned to the bundle. This id will not be reused after the bundle is uninstalled.
<location>	The bundle location string. This is the URL from where the bundle was installed. If this argument contains a blank space or the ″:″ character, for example, ″file:/c:/my file″, then you must enclose this argument in single or double quotation marks (″ ″ or ′ ′).

SMF-defined Commands

This section describes each of the commands that you can use with SMF. The commands are grouped by the following functionality.

- Commands for controlling SMF
- Commands for controlling bundles
- Commands for displaying status
- Miscellaneous commands
- Commands for controlling the console

Commands for controlling SMF

Table 3. Commands for controlling SMF

Command	**Description**
launch	Starts the Service Management Framework. Bundles that were running before the last framework shutdown are restarted. This command has no effect if the framework is already running.
close	Shut down and exit.
exit	Exit immediately (System.exit(0))
gc	Perform a garbage collection. SMF shows the free memory before, the free memory after and the amount of memory gained with the GC.
init	Uninstall all bundles. The framework must be shut down.
shutdown	Shutdown the Service Management Framework. This command has no effect if the framework is already stopped.
setprop <key>=<value>	Set the SMF Property with the given key to the given value.

Commands for controlling bundles

Table 4. Commands for controlling bundles

Command	**Description**
install <url> {start}	Install and optionally start a bundle from the given URL. The URL can be any protocol supported by the Java Virtual Machine. Examples are: • file:/<fully qualified path name to the file> • http://<remote location of the file> • https://<secure remote location of the file> • smfbd:/<bundle name>
uninstall (<id> | <location>)+	Uninstall the specified bundle(s)
start (<id> | <location>)+	Start the specified bundle(s).
stop (<id> | <location>)+	Stop the specified bundle(s)

Table 4. Commands for controlling bundles (continued)

Command	Description
refresh (<id>|<location>)+	Refresh the packages of the specified bundle(s).
update (<id>|<location>)+	Update the specified bundle(s).

Commands for displaying status

Table 5. Commands for displaying status

Command	Description
bundle (<id>|<location>)	Display details for the specified bundle(s). The bundle can be specified with either its bundle id or its location string. The bundle details are the bundle id, the location string, the bundle state and file name, the services registered and used, the packages imported and exported per bundle, and if running with security enabled, the permissions of the bundle.
bundles	Display details for all installed bundles like the bundle command except that it does not list the package and permission information provided by the bundle command.
props	Display System, SMF and SMF Resource Management properties.
threads	Display the current threads and thread groups. If SMF is launched with resource management the memory space currently assigned to the thread is shown as well.
ss	Display the short version of the bundle status. Only the bundle id, bundle type (jar or jxe), bundle state and the bundle name are shown.
status	Display installed bundles and their state and the registered services.
services {filter}	Display registered service details optionally filtered by a string representation of as specified by the OSGi™technology in the interface org.osgi.framework.Filter.
headers (<id>|<location>)+	Print bundle headers from the bundle's MANIFEST.MF file and bundle characteristics from the bundle's IVEATTRS.XML file.
packages {<pkgname>|<id>|<location>}	Display imported/exported package details optionally filtered by package name, bundle id or bundle location string.
log{(<id>|<location>)}	Display log entries optionally filtered by bundle ID or bundle location string. The newest log entry is listed first.

Commands for controlling the console

Table 6. Commands for controlling the console

Command	Description
more	Enable or disable the “--More” prompt to temporarily halt the output in the console window.
disconnect	This command ends the telnet session. This command is only available when SMF is launched with the `-console:port` option

Miscellaneous commands

Table 7. Miscellaneous commands

Command	Description
exec <command>	Execute an OS command in a separate process and wait for its completion.
fork <command>	Execute an OS command in a separate process and return immediately.

Memory Space-related Commands

SMF with resource management adds commands to create, destroy, and inspect memory spaces. These commands are only available if SMFLauncher is launched with the `-console -r[esourcemanagement]` options.

Table 8. Memory space related commands

Command	Description
a <msName> [<o> [<n]]	Allocate a memory space with the specified name and oldSpace and newSpace sizes.
c <msName>	Set the specified memory space as current memory space of the thread that executes the console commands.
d <msName>[, rmsName]	Destroy the specified memory space(s) with remnants space. The memory space will not be removed if there are references to objects in the memory space from objects in other memory spaces and no remnant space has been specified.
lc [<msName>]	Display the classes that have instances in the specified memory space or all memory spaces if none was specified.
lo [<msName>]	Display the objects in the specified memory space or all memory spaces if none was specified.
lp [<msName>]	Display references (pointers) from the specified (or all) memory space(s) to objects that live in other memory spaces.
lr <msName> [,<n>]	Display the references to objects in the specified memory space. Optionally specify how many levels should be shown.
memory	Display memory space status.

Chapter 3. SMF Toolkit Reference

This chapter describes the bundles that are included with the SMF Toolkit. It also provides instruction on how to develop bundles, example applications, and important concepts for developing bundles.

Core Bundles

The bundles included with the toolkit framework provide a collection of basic services, which other bundles can use. Some of these are implementations of services defined by the OSGi™ specification. A subset of core bundles is installed into the SMF Runtime when launched. You can install all of the core bundles from the bundlefiles directory.

The interface specifications of the OSGi standard services are provided in the `OSGi Service Platform Release 2 Service Interfaces` bundle (`osgi-services.jar/osgi-services.jxe`) bundle. The implementations of these services are provided in SMF bundles that are included in this SMF release.

Table 9. Services

Standard Service	Bundle-Name
org.osgi.service.cm.ConfigurationAdmin	Configuration Admin
org.osgi.service.http.HttpService	HTTPService
org.osgi.service.log.LogService	LogService
org.osgi.service.log.LogListenerService	LogService
org.osgi.service.prefs.PreferencesService	Preference Service
org.osgi.service.useradmin.UserAdmin	User Admin

Table 10. Packages

Standard Package	Bundle-Name
org.osgi.service.device	DeviceManager
org.osgi.util.tracker	tracker

These packages and services are documented in the OSGi Service Platform, Release 2 Specification.

The following sections describe the following SMF-specific core bundles:

- OSGi services
- System bundle
- Http service
- Log service
- Device Access
- Preferences service
- Configuraton Admin service
- User Admin service
- Service Tracker

© Copyright IBM Corp. 1999, 2002

- SMFAdmin
- File Admin
- XML Parser APIs
- MicroXML
- XML4J

OSGi™ services

The bundles included with the toolkit framework provide a collection of basic services and packages that other bundles can use. Some of these are implementations of services defined by OSGi specifications.

The interface specifications of the OSGi standard services are provided in the OSGi Service Release 2 Service Interfaces bundle, (for example, the osgi-services.jar or osgi-services.jxe file).

Table 11. Services exported by osgi-services.jar

Standard Service	Bundle-Name	Implementation File Name (*.jar or *_le.jxe or *_be.jxe)
org.osgi.service.cm.ConfigurationAdmin	Configuration Admin	cm
org.osgi.service.http.HttpService	HTTPService	httpservice
org.osgi.service.log.LogService	LogService	logservice
org.osgi.service.LogReaderService	LogService	logservice
org.osgi.service.prefs.PreferencesService	Preference Service	prefs
org.osgi.service.useradmin.UserAdmin	User Admin	useradmin

Table 12. Packages exported by osgi-services.jar

Standard Service	Bundle-Name	Implementation File Name (*.jar or *_le.jxe or *_be.jxe)
org.osgi.service.device	DeviceManager	deviceaccess

Table 13. Packages exported by tracker.jar

Standard Service	Bundle-Name	Implementation File Name (*.jar or *_le.jxe or *_be.jxe)
org.osgi.util.tracker	OSGi Service Tracker	tracker

System Bundle

The toolkit framework defines its system bundle in the `META-INF/SYSTEMBUNDLE.MF` file.

The system bundle registers the following services at startup:

- org.osgi.service.packageadmin.PackageAdmin
- org.osgi.service.permissionadmin.PermissionAdmin

The standard `java.*` packages that are contained in the base class libraries are not included in this list. The ImplType definition is used for this purpose.

Framework

In `smf.jar`, SMF implements the OSGi™ specifications for Framework, Package Admin, and Permission Admin.

Http Service

In `httpservice.jar` and `servlet.jar`, SMF implements the OSGi™ specification for Http Service and provides an HTTP 1.0 Web server with a Java Servlet 2.1 engine. Http Service enables other bundles to dynamically register and unregister servlets and other static resources such as GIF files. You can register HTML files, GIF files, class files, or any resources that can be read through a URL.

handleSecurity plug-in

When you register a resource or a servlet with the Http Service, an Http Context must be provided. The Http Context defines how resources are accessed and how access to the resources is authenticated. The handleSecurity method of Http Context performs the authentication for servlets and resources.

IBM's Http Service implementation supports a plugable default handleSecurity method. This method defines the behavior of the default Http Context handleSecurity method.

Configuring Http Service

Http Service configurations described here are specific to the IBM® implementation of Http Service that is included with SMF.

The IBM implementation of Http Service has several configuration parameters. There are general configuration parameters that affect all of Http Service and there are additional parameters which are used to configure individual ports. These additional parameters allow Http Service to be configured to register multiple Http Service objects, with each object listening on a unique port.

The Http Service manages its configuration parameters with the Configuration Admin Service. The Configuration Admin Service relies on a PID (Persistent Identity) to identify which service to configure. The PID resides in a service property called `service.pid`.

General configuration: These parameters affect all of Http Service and are configured with a Managed Service.

The PID for the Managed Service for Http Service's general configuration is:

`com.ibm.osg.service.http.Http`

Table 14. Http Service General Configuration Parameters

Parameter Name	Java Type	Definition	Valid Values
http.minThreads	`Integer`	Specifies the minimum number of threads in the thread pool.	Valid values are between 0 and 63. The default value is 4.
http.maxThreads	`Integer`	Specifies the maximum number of threads in the thread pool.	Valid values are between 0 and 63. The default value is 20.

Table 14. Http Service General Configuration Parameters (continued)

Parameter Name	Java Type	Definition	Valid Values
http.threadPriority	`Integer`	Specifies the priority of threads in the thread pool.	A valid value is any integer with a valid `Thread` priority. Refer to the Thread class for valid values. The default value is `Thread.NORM_PRIORITY`.

Port configuration: The individual ports are configured with a Managed Service Factory. A Managed Service Factory allows multiple instances of a configuration for a given service. Each configuration that you make under the Managed Service Factory PID directs Http Service to register another Http Service object with the configured parameters.

Note: If no individual configuration exists, then Http Service will not register an Http Service object and therefore, will not listen on any port unless the property, `com.ibm.osg.service.http.defaultports`, is set. Refer to the `smf.properties` file for more information.

The PID for the Managed Service Factory for the individual port configuration is: `com.ibm.osg.service.http.HttpFactory`

Table 15. Http Service Individual instance parameters

Parameter Name	Java Type	Definition	Valid Values
http.port	`Integer`	Specifies the port to listen to for HTTP requests.	The default value is 80.
http.scheme	`String`	Specifies the scheme to use.	Valid values are `http` and `https`. The default value is `http`. **Note:** The evaluation release of SMF does not include support for https.
http.timeout	`Integer`	Specifies the number of seconds to wait before reclaiming a Keep-Alive thread.	Valid values are between 0 and 600. The default value is 30. Specify 0 to disable Keep-Alive support.

Configuring the parameters using SMFAdmin:

Bundles configure the parameters with the Configuration Admin API. You can also configure the parameters with the SMFAdmin User Interface. To configure Http Service using SMFAdmin, under **Bundles** find and click the bundle file with the name **file:bundlefiles/httpservice**. Then, under **Registered Services**, click **org.osgi.service.cm.ManagedService** to modify the general configuration or click **org.osgi.service.cm.ManagedServiceFactory** to modify, add, or delete individual port configurations.

Configuring the HTTPService for multiple ports:

You can configure the HTTPService for multiple ports. Each port configured is represented by a different HttpService registration. An application that uses the HTTPService must register with each port to be accessible on that port.

For example, you might only want to register a Servlet with sensitive data with the HttpService which supports HTTPS on port 443. You might want to register a Servlet that does not contain sensitive data with both the 80 and the 443 port HttpServices.

To ensure that you register a servlet with the correct HttpService, use a **ServiceTracker** with a Filter. A Filter enables your bundle to specify exactly which services you want to know about to the ServiceTracker. Your BundleActivator can extend the **ServiceTracker** class or implement the **ServiceTrackerCustomizer** interface to provide the addingService(), modifiedService(), and removingService() methods. ServiceTracker will call these methods to notify your bundle when these service registration events occur. For example, if your bundle is tracking HttpService registrations, it would be notified when HttpServices are registered and unregistered and your bundle can then use these services accordingly.

The following example shows how to create the Filter and ServiceTracker:

```
//To track any HttpService registrations using port 80.  We would look for the property "http.port
public void start(BundleContext context)
{
        Filter filter = context.createFilter("(&(objectClass=org.osgi.service.http.HttpService)(h
        ServiceTracker httpServiceTracker = new ServiceTracker(context,filter,this);
}
//To track any HttpService registrations using HTTPS.  We would look for the property "http.scheme
public void start(BundleContext context)
{
        Filter filter = context.createFilter("(&(objectClass=org.osgi.service.http.HttpService)(h
        ServiceTracker httpServiceTracker = new ServiceTracker(context,filter,this);
}
```

For an example on how to write a ServiceTrackerCustomizer, see the PizzaServlet Bundle sample source code included in the SMF Toolkit.

For additional documentation, please see the OSGi™ Service Platform release 2 specification.

Log Service

In `logservice.jar`, SMF implements the OSGi™ Log Service specification.

Configuring Log Service

Log Service configurations described here are specific to the IBM implementation of Log Service that is included with SMF. Log Service has a number of configuration options. You can control the log size and the log threshold (Error, Warning, Informational, or Debug severity level). When a threshold level is specified, only entries that are less than or equal to the current threshold are logged. `Error` is the lowest threshold level and `Debug` is the highest level.

The Log Service manages its configuration parameters with the Configuration Admin Service.

The PID for the Managed Service for Log Service's configuration parameters is: `com.ibm.osg.service.log.Log`

Table 16. Log Service Configuration Parameters

Parameter Name	Java Type	Definition	Valid Values
log.size	`Integer`	Specifies the maximum number of entries in the log. Old entries are discarded after the log reaches this number.	Valid values are between 10 and 2000. The default value is 100.
log.threshold	`Integer`	Specifies the lowest level that you want to retain entries in the log. Any level higher than what is specified will not be kept in the log.	Valid values are: LogService.LOG_ERROR LogService.LOG_WARNING LogService.LOG_INFO LogService.LOG_DEBUG

Configuring the parameters using SMFAdmin: Bundles configure these parameters with the Configuration Admin API. You can also configure these parameters with SMFAdmin. To configure Log Service using SMFAdmin, under **Bundles** find and click the bundle file with the name **file:bundlefiles/logservice**. Then, under **Registered Services** click **org.osgi.service.cm.ManagedService** to modify the general configuration.

Device Access

In `deviceaccess.jar`, SMF implements the OSGi™ Device Access specification for the driver manager.

Preferences Service

In `prefs.jar`, SMF implements the OSGi™ Preferences Service specification. For more information on `prefs.jar`, refer to the OSGi specification provided in the `SMFHOME\docs\osgi` directory, or visit www.osgi.org.

Configuration Admin Service

In `cm.jar`, SMF implements the OSGi™ Configuration Admin Service specification. For more information on `cm.jar`, refer to the OSGi specification provided in the `SMFHOME\docs\osgi` directory, or visit www.osgi.org.

User Admin Service

In `useradmin.jar`, SMF implements the OSGi™ User Admin Service specification. For more information on `useradmin.jar`, refer to the OSGi specification provided in the `SMFHOME\docs\osgi` directory, or visit www.osgi.org.

Service Tracker

In `tracker.jar`, SMF implements the OSGi™ Service Tracker specification. For more information on `tracker.jar`, refer to the OSGi specification provided in the `SMFHOME\docs\osgi` directory, or visit www.osgi.org.

SMFAdmin

`smfadmin.jar` is a servlet that provides simple administrative access to SMF. It enables you to manage the life cycle of bundles and view the installed bundles and registered services. You can also use SMFAdmin to configure the IBM implementation of Http Service and Log Service. You can access SMFAdmin at `http://localhost:nn/smfadmin`.

FileAdmin

`fileadmin.jar` is a servlet that provides simple administration access to the file system of the device that hosts SMF. You can access FileAdmin at `http://localhost:nn/fileadmin`.

XML parser APIs

xmlParserAPIs.jar is a library that simply exports the APIs for XML parsing defined by org.W3c.dom for DOM parsing, org.xml.sax for SAX parsing, and javax.xml.parsers for JAXP. The packages exported are:

- org.W3c.dom.html; specification-version=″2.0″
- org.W3c.dom.traversal; specification-version=″2.0″
- org.xml.sax.ext; specification-version=″2.0″
- org.w3c.dom.ranges; specification-version=″2.0″
- org.w3c.dom.events; specification-version=″2.0″
- org.w3c.dom; specification-version=″2.0″
- javax.xml.parsers; specification-version=″2.0″
- org.xml.sax; specification-version=″2.0″
- org.xml.sax.helpers; specification-version=″2.0″

Micro XML

MicroXML.jar provides a very small, lightweight XML parsing service designed to run in the embedded space. In order to achieve its small size, there are trade-offs in function. MicroXML supports SAX version 1.0 and DOM version 1.0. For the most part, it assumes well-formedness, checking for only the most basic errors in form. It is non-validating and non-namespace aware. It does not support DOM entities.

MicroXML registers the javax.xml.parsers.SAXParserFactory and javax.xml.parsers.DocumentBuilderFactory services. These services are an extension of the Java API for XML Parsing (JAXP) 1.1. Bundles wishing to use an XML parser can find one by requesting Service References for these services. Bundles do not have to tie themselves to a particular parser at development time -- they can choose at runtime.

MicroXML is installed as part of the SMF Toolkit, but another larger but more robust parser is available for installation. See “XML4J” for more details.

XML4J

XML4J.jar provides a heavyweight XML parsing service. It supports SAX version 2.0 and DOM version 2.0. It is validating and namespace aware. It is based on the Apache Xerces XML Parser.

XML4J registers the javax.xml.parsers.SAXParserFactory and javax.xml.parsers.DocumentBuilderFactory services. These services are an extension of the Java API for XML Parsing (JAXP) 1.1. Bundles wishing to use an XML parser can find one by requesting Service References for these services. Bundles do not have to tie themselves to a particular parser at development time -- they can choose at runtime.

Creating OSGi™ bundles

This section describes how to create an OSGi™ bundle. For more detailed information about writing bundles, refer to the OSGi Service Platform Release 2 specification.

For an example that demonstrates how to write a bundle, refer to the Pizza servlet bundle that is included with SMF. The Pizza servlet bundle is a simple bundle that registers a servlet with the HTTP Service.

Note: The pizza bundle code resides in the `\samples\source` directory.

Refer to the following sections for information about writing bundles with SMF:

- Understanding OSGi bundles
- Manifest files
- Sample manifest file
- Services
- Conventions for using SMF

Understanding OSGi™ bundles

OSGi™ bundles consist of a jar file that contains Java classes, resources, and a manifest file. Bundles can register services for other bundles to use, use services registered by other bundles, export Java packages for other bundles to use, and import Java packages from other bundles.

Manifest files

Bundles must contain a manifest file. The bundle's manifest file contains data that the framework needs to correctly install and activate the bundle. When you specify data in a manifest file, you must use the headers that were defined by the OSGi™ specification. Because the framework ignores any headers that it does not understand, manifest files do not allow user-defined headers. In addition, you must place the OSGi headers at the beginning of the manifest file. Refer to the OSGi Service Platform Release 2 specification for more information about the OSGi Manifest file format and syntax.

When you create the manifest file for your bundle, you manifest file must be named `Manifest.mf` and must reside in the `META-INF` directory inside the bundle.

Refer to the following list of headers that you will use in your Manifest file.

- Import-Package

  This header specifies any package that a bundle imports from another bundle. If you do not specify the packages you need in this header, you will get a `noClassDefFound` exception.

  Note: You must also specify the package you want to import in the Export-Package header of the bundle that provides the package.
- Export-Package

  If you want your bundle to export its packages so other bundles can import them, you must specify the packages with the Export-Package header. If you do not specify the correct packages, the dependent bundles will not resolve.
- Bundle-Activator

  This is the fully qualified name of the BundleActivator class.

A bundle designates a special class to act as a Bundle Activator. The Framework must instantiate this class and invoke the start and stop methods to start or stop the bundle respectively. The bundle's implementation of the BundleActivator interface enables the bundle to initialize a task, such as registering services, when the bundle starts and to perform clean-up operations when the bundle stops.

Note: BundleActivator is not a mandatory header. If the purpose of your bundle is only to export packages for use by other bundles, then you do not need to specify a BundleActivator.

Refer to the OSGi™ Service Platform Release 2.0 specification for descriptions of other bundle headers, such as the following, which provide bundle description information:

- Bundle-Name
- Bundle-Description
- Bundle-Copyright
- Bundle-Vendor
- Bundle-Version
- Bundle-DocUrl
- Bundle-ContactAddress

Sample manifest file

The following sample manifest file was taken from the Pizza Servlet bundle, which is one of the SMF sample bundles.

```
Import-Package: javax.servlet; specification-version=2.1,
 javax.servlet.http; specification-version=2.1,
 org.osgi.framework; specification-version=1.0,
 org.osgi.service.http; specification-version=1.1,
 org.osgi.service.log; specification-version=1.0,
 org.osgi.util.tracker; specification-version=1.0
Bundle-Activator: com.ibm.osg.sample.pizza.PizzaBundle
Bundle-Name: Pizza Servlet
Bundle-Description: A demonstration bundle containing a simple servlet
Bundle-Copyright: Licensed Materials - Property of IBM. (C) Copyright IBM
 Corp. 2000-2001 All Rights Reserved. IBM is a registered trademark of IBM Corp.
Bundle-Vendor: IBM Pervasive Computing
Bundle-Version: 3.0
Bundle-DocUrl: http://www.ibm.com/pvc/
Bundle-ContactAddress: pervasive@us.ibm.com
```

Services

In the OSGi™ environment, bundles are built around a set of cooperating services that are available from a shared service registry. The service interface defines the OSGi service, which is implemented as a service object.

Registering and unregistering a service with the OSGi Framework

When the framework invokes your BundleActivator's start method, it passes a BundleContext object to your bundle. You bundle can use the BundleContext object to interact with the framework by calling the BundleContext object's methods. One method that you bundle calls is `registerService`, which uses a service object and an interface name to register a service with the framework's service registry.

In the following example, a service called `com.ibm.osg.example.mtservice.MyTestService` registers with the framework. This service prints a debugging message to standard out.

The following is the interface that our service registers with. Ideally, you should create one package that contains all of the classes and interfaces that will be exported to the OSGi framework. For our example, the `com.ibm.osg.example.mtservice` package will be exported.

```
/******* START OF com/ibm/osg/example/mtservice/MyTestService.java *******/
package com.ibm.osg.example.mtservice;
public interface MyTestService {
 // One method is provided by the service.  This method will simply print
 // the message to standard out.
 public void printMessage(String message);
}
/******* END OF com/ibm/osg/example/mtservice/MyTestService.java *******/
```

The following is the class that will provide the implementation for our service. The class is not a part of the exported package `com.ibm.osg.example.mtservice`. Ideally, you should not export the packages that contain classes that provide implementation to services.

```
/****** START OF com/ibm/osg/example/mytestservice/MyTestService.java ******/
package com.ibm.osg.example.mytestservice;
public class MyTestService implements com.ibm.osg.example.mtservice.MyTestService{
 public void printMessage(String message){
  System.out.println("MyTestService - " + message);
 }
}
/****** END OF com/ibm/osg/example/mytestservice/MyTestService.java ******/
```

The following is the BundleActivator class that will register the `com.ibm.osg.example.mtservice.MyTestService` service with the framework.

```
/****** START OF com/ibm/osg/example/mytestservice/MyBundleActivator.java ******/
package com.ibm.osg.example.mytestservice;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceRegistration;

public class MyBundleActivator implements BundleActivator {

ServiceRegistration registration;

/*Create a new instance of the TestService and then use the BundleContext object to register it.*\
/*Store the registration object to use to unregister the service when the bundle is stopped by the framework.*\
 public void start(BundleContext context) {
 MyTestService testservice = new MyTestService();
   if( registration == null ){
    registration = context.registerService("com.ibm.osg.example.mtservice.MyTestService",testservice,null);
   }
 }
 public void stop(BundleContext context) {
  if ( registration != null ){
   registration.unregister();
  }
  registration=null;
 }
}
/****** END OF com/ibm/osg/example/mytestservice/MyBundleActivator.java ******/
```

Getting and un-getting services from the OSGi Framework

Bundles register and unregister services. Because bundles are volatile, bundles that depend on services must account for the possibility that the requested service might not be present. The service can be registered or unregistered at any time. You can use a ServiceTracker to enable your bundles to query or listen for service registrations and to react accordingly.

```
/****** START OF com/ibm/osg/example/mygetservice/MyBundleActivator.java ******/
package com.ibm.osg.example.mygetservice;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.util.tracker.ServiceTracker;
import com.ibm.osg.example.mtservice.MyTestService;

public class MyBundleActivator implements BundleActivator, Runnable{
 private boolean done=false;
 private ServiceTracker testServiceTracker;

 // Bundle Activator Start Method
 public void start(BundleContext context){
/* Here we initialize and open our ServiceTracker. It will track any service registering under
the "com.ibm.osg.example.mtservice.MyTestService" interface. *\
  testServiceTracker = new ServiceTracker(context,"com.ibm.osg.example.mtservice.MyTestService",null);
  testServiceTracker.open();
// Here we start a thread that will continue to use the our service until
// the bundle is stopped.
  Thread t=new Thread(this);
  t.setName("mygetservice thread");
  t.start();

 }
/*Bundle Activator Stop Method -- here we stop the thread and close the
ServiceTracker*\

  public void stop(BundleContext context){
   done=true;
   testServiceTracker.close();
  }
//Here is a method that uses the service we are tracking.  First we get
//the service from the tracker, then we call it's printMessage method.
  public void useService(String message){
   MyTestService testService = (MyTestService)testServiceTracker.getService();
   if( testService != null ){
    // If the service is available then use it.
    testService.printMessage(message);
   }
   else{
    // If the service is not available then perform an acceptable action.
    // Here we just print the message to standard out and indicate the service
    // was not available.
    System.out.println("No MyTestService available - " + message);
   }
  }

// Simply continues to use the test service every second until the done flag is set.
  public void run(){
   int i = 0;
   done = false;
   while (!done){
    useService("message from test " + i++);
    try{
     Thread.sleep(1000);
    }
     catch( InterruptedException ie ){
    }
    }
   }
}
/****** END OF com/ibm/osg/example/mygetservice/MyBundleActivator.java ******/
```

For an example of using ServiceTrackers and getting services, refer to the Pizza servlet sample source code (`PizzaBundle.java`), which is located in `SMFHOME\samples\source\src-pizza.zip`.

Conventions for creating bundles

When you create SMF bundles, use the following conventions:

- Clean up objects and threads properly during your stop method. SMF does not terminate lingering threads.
- Do not use a lot of time for your bundle's start and stop methods. Services are started and stopped one at a time. If your start or stop method uses a lot of time, it prevents other services from stopping and starting.
- Do not assume that a service will always be present. The bundles that register these services might not be available at all times.

Sample applications

Sample bundles are located in the `SMFHOME\samples` directory. You can use the sample code to develop your own bundles. The following sections include high-level descriptions of the samples:

- Pizza servlet
- Handle security
- Simple permission policy
- Driver locator sample
- Calendar sample

Pizza servlet

The Pizza servlet uses the Http Service to show the results of input from radio buttons and the output of a calculation viewed through a browser. This sample also demonstrates the use of the ServiceTracker and ServiceTrackerCustomizer. You can access this sample at `http://localhost:`*`nn`*`/pizza`.

Handle Security

This sample demonstrates the use of the User Admin Service and the Service Tracker.

The handleSecurity sample provides an example of a default handleSecurity plug-in. The handleSecurity method implements basic authentication and uses the UserAdmin Service to validate user names and passwords. After you have installed this plug-in, any resource or servlet registered with the default Http Context will be authenticated using the handleSecurity method of the plug-in.

This sample provides an implementation of the handleSecurity method for the Default Http Context in the IBM Http Service. Both smfadmin and fileadmin use the Default Http Context. If this sample is installed and active, which is the default state of the SMF Toolkit, the sample will authenticate access to smfadmin and fileadmin using the User Admin Service to authenticate the supplied user name and password. If this sample is not active (because it has been stopped or uninstalled), then access to smfadmin and fileadmin will not be authenticated.

Simple Permission Policy

This sample uses the Permission Admin service to configure the permission of a bundle. This sample reads the META-INF/permission.info resource in each installed bundle and calls Permission Admin to set those permissions for the bundle.

Simple Sample

The simple sample demonstrates how a BundleActivator that sends a log message to the standard OSGi™ LogService when started and stopped. If the LogService is not available then the message is written to `System.out` instead.

Driver Locator Sample

A DriverLocator returns a list of driver IDs based on a given device's properties, and it returns the input stream for a given driver ID. The following sample maps a property name to a value, driver ID and location in an XML input file. You can map multiple DriverIDs to a single property name/value pair. Following is a sample device.properties configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
 <driverlocator>
   <devicepropertypair property="FooAppliances" value="Oven">
     <driveridlocation driverID="343Oven" driverLocation="http://www.foo.com/drivers/343Oven" />
      <driveridlocation driverID="293Oven" driverLocation="http://www.foo.com/drivers/293Oven" />
   </devicepropertypair>
   <devicepropertypair property="FooAppliances" value="Toaster">
   <driveridlocation driverID="400Toaster" driverLocation="http://www.foo.com/drivers/400Toaster"
   </devicepropertypair>
 </driverlocator>
```

This file will be stored in the SimpleDriverLocator bundle.

Calendar sample

This sample bundle provides an example of how to use the jdbc related services provided by other bundles. The sample also demonstrates the use of various proven design patterns in the developing of a servlet 2.1-based application.

The overall application follows the Model-View-Controller (MVC) design pattern. This design pattern divides the application into distinct parts, which enables you to maintain, enhance, and expand the code. The three parts are:

- Model - consists of a business logic, databases, and so on.

  The model portion of the code utilizes the jdbc interface and stores its data in the DB2e™ database. A trial version of DB2e is available as a bundle, along with a database_enabler bundle, which provides the java.sql.* interface. The calendar servlet illustrates how to set up the Manifest file to reference the services provided by these other bundles.
- View - consists of the text, graphics, links, forms, and various layout related items.

  The view portion of the code is targeted toward a 1/4 VGA screen with only HTML 3.0.2 support. Since the view is separated from the model, other views could be provided without any effect on the model code - easing the maintenance burden.
- Controller - consists of the services as the linkage between the model and the view.

  The controller implementation illustrates several design patterns: including the ″Front Controller″ patter and the ″Command Controller Pattern″.

Platform

SMF provides an abstraction between the OSGi™ framework and the underlying platform where the framework runs. SMF uses the Platform interface described in `\docs\smf`. The Platform interface provides SMF with a platform specific implementation of some framework APIs as well as persistent storage for bundles and permission configurations.

Default platform

SMF includes a DefaultPlatform class, which enables SMF to run on any standard platform that includes a file system. You can write a custom Platform implementation for SMF to use for other non-standard platforms, such as platforms that do not include a file system.

SMF Memory Considerations

When running SMF on the J9 VM you must consider the following special issues regarding memory usage:

- Memory allocation: The extra memory that is allocated by the VM per bundle. Refer to "Memory Allocation" for more information.
- Memory deallocation: When this extra memory is freed. Refer to "Memory Deallocation" on page 30 for more information.

Memory Allocation

Each bundle has its own class loader allocating memory segments of two types.

Table 17. Memory Segment Types

Segment Type	**Description**
ROM classes	ROM Classes contain the read-only part (i.e., code) of the classes loaded by a class loader. All memory segments of this type are allocated in the same size as defined with the `-Xmco:<n>` option (n * 1024) if they hold classes loaded from JAR bundles. If a segment fills up, then another segment of the same size and type is created. "ROM Classes" memory segments that hold classes from JXE bundles on the other hand are allocated in the size of the rom.classes file in the JXE file. In case SMF is configured to use the Flash Manager to store the bundles in flash memory then the J9 VM just points to the memory location in of rom.classes file in the flash and therefore does not require any extra RAM for the ROM portion of the classes.
RAM classes	"RAM Classes" contain the space to hold the pointers to the super class, implemented interfaces, static objects and other data for all classes loaded by a class loader. The size of all the memory segments of this type are set with the `-Xmca:<n>` option (n * 1024). If a segment fills up then another segment in the same size and type is created for the class loader.

The defaults for the `-Xmca` and `-Xmco` options are different from platform to platform, in many cases it is set to 1 MB. The default sizes might be far to too high for your application. If there are many bundles installed and used then the available memory is used up quickly.

Use MicroAnalyzer to determine the best size of either segment type for your application.

Other Memory Segments

Other memory segments to consider are the "NewSpace" (`-Xmn<n>`), where the new objects are allocated, and the "OldSpace", which holds "long lived" objects. The "NewSpace" segment is fixed size, but take into account that there is another memory segment of the same size as the "NewSpace" called the "SurvivorSpace". The "OldSpace" segment grows when the garbage collector cannot free enough space to hold new objects as set with the `-Xmoi<n>` parameter. If <n> is 0 then an OutOfMemoryError is thrown.

Memory segments used by the Garbage Collector are the "Remembered Class Set" and "Remembered Object Set" (`-Xmr<n>` for both) and the "IGC Scan Queue" (`-Xgc:agcqs=<n> (n *8)`). All these memory segments must fit into the memory reserved with -Xmx<n> option.

The following is an example of how the J9 VM's memory options can be used:

```
REM Set J9 memory segment sizes to simulate the target device
J9-Xmca8k -Xmco8k -Xmo2m -Xmoi0 -Xmn256k -Xmx8m -Jxe:smf.jxe
```

Memory Deallocation

When a bundle is uninstalled or updated* the bundle class loader's memory segments are freed automatically if there is not another reference to any type or to an object that extends or implements any type. *If a bundle is updated then a new bundle class loader is created with a new set of memory segments. Object instances are "garbage collected" when there is no more reference to the object from any other object, from a class loaded by another class loader, or from a VM internal data structure like the thread stacks and the finalize queue. There is the potential that the "ROM Classes" and "RAM Classes" memory segments of a bundle class loader are never freed even though the bundle was uninstalled a long time ago. For example a bundle defines its own Exception class and appends an instance of this Exception in a log message and sends it to the LogService bundle. If the LogService collects all log entries in an internal data structure then the LogService references the Exception and the bundle's class loader memory segments cannot be freed.

Resource Management

You can enable resource management using the `-resourceManagement` launch option. When you enable resource management SMF manages the following resources:

- The bundle storage space. For example a bundle will not be installed if not enough flash memory is available to store it. The required amount of bundle storage is defined by the size of the bundle's JAR or JXE file.
- The amount of object memory that a bundle gets when it is started and which it cannot exceed. A bundle is assigned to its separate memory space where its threads will allocate their objects. If the memory space is full and the garbage collector cannot release any objects the thread(s) will receive an OutOfMemoryError. For more information about memory spaces, refer to the class libraries documentation.
- Timeout values for the bundle activator's start and stop calls. A bundle activator running in an endless loop will not block the framework's master thread.
- The maximum number of threads for the whole device and per bundle. In an embedded device threads are an expensive commodity. A bundle that creates too many threads can easily render a device useless.
- The maximum number of sockets for the whole device and per bundle.
- The maximum number of files and directories in the bundle's private data area (as available to the bundle with BundleContext.getDataFile(" ")) and the total size of these files.
- Trace resource consumption, like socket and file I/O.

For more information on managing resources, refer to the following sections:

- smfres.properties file
- Enabling and disabling resource management
- Trace
- Memory spaces
- Threads
- Socket management
- File system management

smfres.properties file

The resource `smfres.properties` contains the parameters that control the resource manager of the Service Management Framework. These are the defaults and maximum values for the bundles and the device.

You can copy the `smfres.properties` resource to the working directory of the toolkit environment. If an smfres.properties file resides in the current working directory, it supersedes the resource in the class path.

The `smfres.properties` resource also contains parameters to enable or disable the management of the various resources. If all resource management is disabled then the system will behave exactly like the standard Service Management Framework, except that the footprint is a little larger and slightly slower due to some extra checking and method calls.

If the resourcemanagement launch option is not set, then the SMF Resource manager is not installed and the smfres.properties file is not used. If the resourcemanagement launch option is set, then the SMF Resource Manager reads the smfres.properties file and adds them to the System properties. If the com.osg.smf.reman.enabled property is false, no memory spaces are created and the resource limits of the bundle are not checked; however, if is possible to enable the trace functionality on the different resources and to use the commands provided by the Resource Manager console extensions.

Enabling and disabling resource management

The `smfres.properties` file contains a parameter to enable or disable the resource management. If the resource management is disabled, then the system will behave exactly like the standard SMF, except that the footprint is a little larger and slightly slower because of some extra checking and method calls.

The resource management parameter starts with `com.ibm.osg.smf.resman`

Table 18. Parameter Suffixes Related to Enabling and Disabling Resource Management

Parameter suffix	Description	Default
`.enabled`	Enables resource management. If this parameter is set to false, all other parameters in the following tables are not applicable.	true

Trace

Enable the tracing of resource consumption by bundles, like the creation of threads and sockets, socket and file I/O, and memory space usage.

Trace parameters start with `com.ibm.osg.smf.resman.trace`

Table 19. Parameter Suffixes Related to Tracing

Parameter suffix	Description	Default

`.memoryspcae.creation`
`.memoryspace.removal`
`.memoryspace.switching`
`.thread.creation`
`.socket.creation`
`.socket.input`
`.socket.output`
`.file.output`
	Individual parts of the system that can be traced	false

`.file`	The file name for the trace output. If not specified, then the trace is written to system.out.	system.out

Memory Spaces

Service Management Framework resource management uses the functionality that is provided by memory spaces, which are installed with the Service Management Framework. This implementation sets up a memory space for every resolved bundle if so instructed in the `smfres.properties` file. Bundle groups are not yet supported.

Note: For more information about memory spaces, refer to the class libraries documentation.

Parameters

All memory-related parameters begin with `com.ibm.osg.smf.resman.memory`.

Table 20. Parameter Suffixes Related to Memory

Parameter suffix	Description	Default
`.default.oldspacesize`	The default OldSpace memory segment size for a bundle that does not specify it in the bundle's resource requirements file.	128000
`.default.newspacesize`	The default NewSpace memory segment size for a bundle that does not specify it in the bundle's resource requirements file.	128000
`.default.enforced`	Use default memory space sizes for all bundles, regardless of the specification in the bundle's resource requirements file.	false
`.remnantsspace.name`	The remnants memory space (RMS). When a memory space is removed the objects which have references from objects in other memory spaces are moved to the RMS. Objects in the NewSpace of the memory space are moved to the RMS NewSpace and objects in the OldSpace of the memory space are moved to the RMS OldSpace. If `com.ibm.osg.smf.resman.memoryspace.remnantsspace.name` is empty then no remnants memory space will be created and the remove will fail if there are references to objects in the memory space that is to be removed. Note that the name "Base" is reserved for the base memory space. In this	Base
`.remnantsspace.oldspacesize`	The size of the OldSpace memory segment of the remnants memory space.	128000
`.remnantsspace.newspacesize`	The size of the NewSpace memory segment of the remnants memory space.	128000

Multiple Memory Spaces

Some bundles need to be aware that they run in a system with multiple memory spaces. For example:

- A bundle (like the HttpService) that creates a thread on behalf of another bundle (servlet) must make sure that this thread is assigned to the memory space of the servlet, otherwise the Http service itself would be charged with all objects that are created by the servlet.
- The user interface shell (like the SampleShell) that manages applications provided by other bundles must make sure that its thread is assigned to the memory space of the currently active application.
- A bundle that does not create objects, like a Service definition bundle that only contains interfaces, can specify 0 as the size for their NewSpace and their OldSpace memory segments.
- A bundle that only exports packages, like Bundle Class Libraries (IveLib), needs to specify an appropriate amount of NewSpace or better OldSpace to accommodate the static objects of its classes. Use the MicroAnalyzer tool to determine the proper memory space segment sizes.
- A bundle that creates a thread should check for the toolkit exception `com.ibm.ive.eccomm.osg.smf.ResourceException` or any of its superclasses if the thread creation was allowed by the resource manager.

For more information about memory spaces, refer to the class libraries documentation.

Threads

Besides managing the maximum number of threads of the device and per bundle, the resource manager can also terminate a thread safely by moving it to the DeathRow memory space, a memory space with no NewSpace and an OldSpace with no extra space for objects.

All thread related parameters start with `com.ibm.osg.smf.resman.threads`.

Table 21. Parameter Suffixes Related to Threads

Parameter suffix	Description	Default
`.max`	The maximum number of user threads that can be active at any one time on this device. These are all the threads that belong to the 'SMF-Master' thread group or any of its children groups, excluding the threads that start or stop bundle's asynchronously. No check is done if -1 is specified.	-1
`.bundle.max`	The maximum number of threads a bundle can create at any one time regardless of what is specified in the bundles' IVERES.XML file. No check is done if -1 is specified.	-1
`.bundle.max.default`	The maximum number of threads a bundle can create at any one time, if there is no "Threads" entry i the bundle's IVERES.XML file and `com.ibm.osg.smf.resman.threads.bundle.max = -1`. No check is done if -1 is specified.	-1
`.bundle.max.priority`	The maximum priority of a bundle's thread group.	10
`.bundle.start.priority`	Bundles are started asynchronously with a thread with this priority. The thread priority cannot be higher then the maximum priority (com.ibm.ive.res.threads.bundle.max.priority). Threads created by bundles will default to this priority.	5
`.bundle.start.timeout`	The timeout value assigned to the bundles start thread. If the timeout exceeded then the start thread is terminated. No check is done if 0 is specified.	0
`.bundle.stop.priority`	Bundles are stopped asynchronously with a thread with this priority. The thread priority cannot be higher then the maximum priority (com.ibm.ive.res.threads.bundle.max.priority). Threads created by bundle's stop() routine will default to this priority.	5
`.bundle.stop.timeout`	The timeout value assigned to the bundles stop thread. If the timeout is exceeded, then the stop thread is terminated. No check is done if 0 is specified.	0

Socket management

All socket related parameters start with `com.ibm.osg.smf.resman.sockets`.

Table 22. Parameter Suffixes Related to Sockets

Parameter suffix	Description	Default
`.max`	The maximum number of sockets that can be created by all bundles at any one time on this device. No check is done if -1 is specified.	-1
`.bundle.max`	The maximum number of sockets a bundle can open at any one time regardless of what is specified in the bundles' IVERES.XML file. No check is done if -1 is specified.	-1
`.bundle.max.default`	The default number of sockets a bundle can open at any one time if there is no ″Sockets″ entry in the bundle's IVERES.XML file and com.osg.smf.resman.sockets.bundle.max.	-1

File system management

All file system related parameters start with `com.ibm.osg.smf.resman.files`.

Table 23. Parameter Suffixes Related to Files

Parameter suffix	Description	Default
`.bundle.max`	The maximum number of files and directories a bundle can create in it's private data area (BundleContext.getDataFile(String)) regardless of what is specified in the bundle's IVERES.XML file. No check is done if -1 is specified.	-1
`.bundle.max.default`	The default number of files and directories a bundle can create in its private data area (BundleContext.getDataFile(String)) if there is no ″Files″ entry in the bundle's IVERES.XML file and com.osg.smf.resman.sockets.bundle.max = -1. No check is done if -1 is specified.	-1
`.bundle.quota`	The maximum size of all files a bundle can create in its data area regardless of what is specified in the bundles' IVERES.XML file. No check is done if -1 is specified.	-1
`.bundle.quota.default`	The default maximum size of all files a bundle can create if there is no ″Quota″ entry in the IVERES.XML file and `com.ibm.osg.smf.resman.files.bundle.quota = -1` No check is done if -1 is specified.	-1

Chapter 4. Using the SMF Toolkit

Managing a toolkit

You can manage the SMF Toolkit from a command line. You can perform the following management tasks:

- Starting the SMF Toolkit
- Showing runtime properties
- Closing the SMF Toolkit
- Launching the framework
- Shutting down the framework

Starting the SMF Toolkit

You can start the SMF Toolkit from the command line with several options. The options are:

- `-ide`
- `-console`
- `-resourceManagement`
- `-platform`

If you want to manage the SMF Toolkit from the command line, you must launch SMF with the -console option. See "Starting the SMF Toolkit from the command line" on page 7 for more information.

Showing runtime properties

You can view and modify the current values of the properties of a running client. Thes properties consist of the System properties and the smf properties. The `props` command enables you to view the current property values. The `setprop` command enables you to set the value of any property.

The `props` command takes no parameters.

The following example of the props command sets the default port property. There must not be any spaces between the key and value. If the property is set, it will be repeated back as shown below:

```
smf> setprop org.osgi.service.http.port=80
Setting Properties:
        org.osgi.service.http.port=80
```

Closing the SMF Toolkit

You can close the SMF Toolkit using any of the following command line options:

- close - shutdown and exit
- exit - exit immediately (System.exit(0))

See "SMF-defined Commands" on page 11

© Copyright IBM Corp. 1999, 2002

Launching the framework

You can launch the framework from the SMF Toolkit command line with the `launch` command. This attempts to start all bundles that were running before the framework was last shutdown. See "SMF-defined Commands" on page 11.

Shutting down the framework

You can shut down the framework without exiting with the `shutdown` command. "SMF-defined Commands" on page 11.

Managing bundles

If you launched SMF with the `-console` option, you can manage the bundles in the SMF Toolkit from within the SMF console. You can perform the following management tasks:

- "Installing a bundle"
- "Starting a bundle"
- "Stopping a bundle" on page 39
- "Updating a bundle" on page 39
- "Uninstalling a bundle" on page 39
- "Viewing bundle properties" on page 40
- "Viewing service properties" on page 40

Installing a bundle

You can install a bundle from the SMF console using the `install` console command. You must specify the protocol to use and the bundle to be installed.

Following are examples of the `install` command:

- The following example installs mybundle.jar from the specified location on the local file system:

  ```
  smf> install file:/c:/bundlefiles/mybundle.jar
  ```
- The following example installs mybundle.jar from the local file system and marks it to start when the framework is launched:

  ```
  smf> install file:/c:/bunlefiles/mybundle.jar start
  ```
- The following example installs yourbundle.jxe from the specified URL:

  ```
  smf> install http://bundlesite.com/yourbundle.jxe
  ```

Starting a bundle

You can start a bundle that is in INSTALLED or RESOLVED state. The framework will attempt to resolve all dependencies and start the bundle. If all dependencies cannot be resolved, the bundle will fail to start and it will stay in RESOLVED state.

To start a bundle, use the `start` console command. You must specify the bundle number or bundle location. See "SMF-defined Commands" on page 11.

The following are examples of using the `start` command:

- The following example starts the bundle with the ID of 5:

  ```
  smf> start 5
  ```
- The following example starts the bundle with the specified location:

  ```
  smf> start file:bundlefiles/mybundle.jar
  ```

Stopping a bundle

You can stop a bundle that is in ACTIVE state with the `stop` console command. You must specify the bundle number or bundle location. See “SMF-defined Commands” on page 11.

Following are examples of using the `stop` command:

- The following example stops the bundle with the ID of 5:
  ```
  smf> stop 5
  ```
- The following example stops the bundle with the specified location:
  ```
  smf> stop file:bundlefiles/mybundle.jar
  ```

Updating a bundle

You can request that a bundle be updated with the `update` console command. You must specify the bundle number or bundle location. The bundle will be stopped, replaced with a new bundle if one exists, and restarted.

Following are examples of using the `update` command:

- The following example updates the bundle with the ID of 5 from the location from which it was installed:
  ```
  smf> update 5
  ```
- The following example updates the bundle with the specified location from which it was installed:
  ```
  smf> update file:bundlefiles/mybundle.jar
  ```
- The following example updates all bundles from the locations from which they were installed:
  ```
  smf> update *
  ```

You can request that the packages that a bundle imports be refreshed with the `refresh` console command. You must specify the bundle number or bundle location. See “SMF-defined Commands” on page 11.

Following are examples of using the `refresh` command:

- The following example refreshes the bundle with the ID of 5:
  ```
  smf> refresh 5
  ```
- The following example refreshes the bundle with the specified location:
  ```
  smf> refresh file:bundlefiles/mybundle.jar
  ```

Uninstalling a bundle

To uninstall a bundle, use the `uninstall` console command. You must specify the bundle number or the bundle location. The bundle will be removed from the framework. See “SMF-defined Commands” on page 11.

Following are examples of using the `uninstall` command:

- The following example uninstalls the bundle with the specified bundle ID and any prerequisite bundles that are no longer needed:
  ```
  smf> uninstall 5
  ```
- The following example uninstalls the bundle with the specified location and any prerequisite bundles that are no longer needed:
  ```
  smf> uninstall file:bundlefiles/mybundle.jar
  ```

Viewing bundle properties

You can view the properties of a bundle with the `bundle` console command. You must specify the bundle number or bundle location.

Following are examples of using the `bundle` command:

- The following example displays information for the bundle with the ID of 5:

```
smf> bundle 5
```

- The following example displays information for the bundle with the specified location:

```
smf> bundle file:bundlefiles/mybundle.jar
```

You can view the properties of all bundles with the `bundles` console command. This command takes no arguments.

You can view the bundle's manifest headers, IVEATTRS.XML and IVERES.XML file with the `headers` console command. You must specify the bundle number or bundle location. See "SMF-defined Commands" on page 11.

Following are examples of using the `headers` command:

- The following example displays header information for the bundle with the ID of 5:

```
smf> headers 5
```

- The following example displays header information for the bundle with the specified location:

```
smf> headers file:bundlefiles/mybundle.jar
```

Viewing service properties

You can view the properties of one or more services with the `services` console command. The default is to show all services properties but you can optionally specify a filter to reduce the output. The filter is of the form (service.property=value).

Following are examples of using the `services` command:

- The following example displays information for all services:

```
smf> services
```

- The following example displays information for all services whose service.vendor property value = ″IBM″:

```
smf> services (services.vendor=IBM)
```

See "SMF-defined Commands" on page 11.

Chapter 5. Service Management Framework Bundle Requirements

Bundles can be managed by an SMF Bundle Server, which is part of WebSphere Device Developer 5.0.

In order for a bundle to be fully managed by a Service Management Framework bundle server, additional information about the bundle is required.

- The bundle must specify the specification-version for every package that it exports.
- The bundle must include an `IVEATTRS.XML` file.

The bundle can also include an `IVERES.XML` file.

In an SMF Toolkit environment, `IVEATTRS.XML` and `IVERES.XML` are ignored.

Specification Version

All bundles must express a specification-version for all exported packages. This is because this information is required by the algorithm which finds the set of all prerequisite bundles for a given bundle.

Without this information, the bundle server will be unable to ensure that a package exported by one bundle is adequate to support another. It will also be unable to tell if a bundle already on the Runtime needs to be updated or not.

© Copyright IBM Corp. 1999, 2002

IVEATTRS

The IVEATTRS.XML file is used by the bundle server in selecting the best bundle for a particular platform. A JXE bundle, for example, would be inappropriate for a non-J9 VM.

Note: A JXE bundle is an optimized JAR format that contains prelinked Java classes.

Similarly, if a JXE bundle has been created for a little-endian architecture, it should not be provided to a Runtime running on a big-endian processor; all the words and long words would be the wrong endianness. The IVEATTRS.XML file is placed within the META-INF directory. The file name must be in all upper-case letters. An example is shown here.

```
<?xml version="1.0" standalone="yes"?>
<IVEAttrs>
   <Processor/>
   <Endian>le</Endian>
   <AddressLength>32</AddressLength>
   <OS/>
   <OSVersion/>
   <VM>J9v14</VM>
   <ImplType>JCL_Gateway_1_3</ImplType>
   <Language>en</Language>
   <Country>us</Country>
   <Replaces/>
</IVEAttrs>
DTD for IVEATTRS.XML Files
<!-- Name: -//com-ibm-ive-eccomm//DTD IVEATTRS//EN -->
<!-- Version: 1.0 4/12/2000 -->
<!ELEMENT IVEAttrs (Processor,OS,OSVersion,VM,ImplType,
    Language,Country,Replaces,RAMRequirements)>
<!-- Processor identifies the target's architecture. Wherepossible draw from OSD specification-->
<!ELEMENT Processor (#PCDATA)>
<!-- OS identifies the target's operating system. Where possible draw from OSD specification-->
<!ELEMENT OS (#PCDATA)>
<!-- OSVersion identifies the version of the target's operating system.-->
<!ELEMENT Endian (#PCDATA)>
<!-- Endian identifies whether the client is operating as a big or little endian processor-->
<!ELEMENT AddressLength (#PCDATA)>
<!-- AddressLength specifies the length of the target processor architecture's address-->
<!ELEMENT OSVersion (#PCDATA)>
<!-- VM identifies the JVM running on the target-->
<!-- ImplType is a code which identifies the base Java class
    libraries available on the client-->
<!ELEMENT ImplType (#PCDATA)>
<!-- Language specifies the target's ISO 639 language code-->
<!ELEMENT Language (#PCDATA)>
<!-- Specifies the target's A2 ISO 3166 country code-->
<!ELEMENT Country (#PCDATA)>
<!-- Interpretation of the Replaces tag is reserved-->
<!ELEMENT Replaces (#PCDATA)>
```

- Implementation type
- Compatible implementation types

Implementation Type

The ImplType tag is used to identify the base class library support required by the bundle. Since the base configuration will vary greatly amongst targets of differing capabilities, there is no predefined list of acceptable values for this attribute. If a new custom base configuration is defined for a new platform, it must be assigned a new implementation type code.

Compatible Implementation Types

Some base configurations will be compatible with others. For example, a bundle designed for jclGwp can reasonably be expected to run on jclMax. jclGwp is said to be compatible with jclMax. The bundle server must keep a list for each ImplType of those ImplTypes with which it is compatible. Having such a list will allow the bundle server to determine that a bundle targeted at jclGwp is also appropriate for a Runtime running jclMax. All things being equal, if an exact match for ImplType is available it should be chosen over a different but compatible ImplType.

Making the bundle server aware of compatible implementation types allows developers to make bundles available to multiple target types without having to create multiple bundles which differ only in their ImplTypes.

IVERES

A bundle developer uses the IVERES.XML file to record a bundle's expected impact on its host's resources. The bundle server can use this information to ensure that bundles are not provided to Runtimes with insufficient resources to run them. The Runtime framework may use the IVERES.XML file to enforce resource constraints.

Like IVEATTRS.XML, the IVERES.XML file is placed in the META-INF directory. The filename must be all uppercase.

The format of the IVERES.XML is open-ended to allow specification and management of resource requirements for resources not yet identified. A typical IVERES.XML file might look like the following:

```
<?xml version="1.0" standalone="yes"?>
<IVERes>
   <Resource>
      <Name>RAM</Name>
         <Requirement>19507</Requirement>
   </Resource>
   <Resource>
      <Name>Threads</Name>
         <Requirement>6</Requirement>
   </Resource>
</IVERes>
DTD for IVERES.XML Files
The DTD for the IVERES.XML File is:
<!-- Name: -//com-ibm-ive-eccomm//DTD IVERES//EN -->
<!-- Version: 1.0 4/5/2000 -->
<!ELEMENT IVERes (Resource*)>
<!ELEMENT Resource (Name,(Requirement|Available))>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Requirement (#PCDATA)>
<!ELEMENT Available (#PCDATA)>
```

Refer to "Defined Resource Names" on page 44 for more information.

Defined Resource Names

The following resource names are reserved:

Table 24. Reserved Resource Names

Resource Name	**Description**
NewSpaceSize	The number of bytes to be allocated in each new space region of the bundle's memory space
OldSpaceSize	The number of bytes to be allocated for tenured objects in the bundle's memory space
Files	The total number of files a bundle can create in its data directory.
Quota	The number of bytes of storage within the local filesystem required by the bundle
Sockets	The maximum number of socket connections the bundle may have open at any one time
Threads	The number of threads the bundle may create at any one time

Appendix. Service Management Framework Files

The following tree shows the directory of SMF.

```
bundle.jxeLinkOptions
bundlefiles/
bundlesdata
docs/
flashplatform.jar
jarbundles/
jxebundles/
makejxes
makejxes.bat
osgi.jar
resmanconsole.jar
resmanimpl.jar
samples/
servlet.jar
smf
smf.bat
smf.jar
smf.jxeLinkOptions
smf.policy
smf.properties
smfbd.jar
smfconsole.jar
smfjdk
smfjdk.bat
smfres.properties
smftoolkit.readme
smf-foundation.jxeLinkOptions
smf-foundation_be.jxe
smf-foundation_le.jxe
smf-rm.jxeLinkOptions
smf-rm_be.jxe
smf-rm_le.jxe
```

A brief description of the contents of the SMF follows, in alphabetical order:

bundlefiles/
The **bundlefiles** folder is the source directory that contains a collection of bundles which may be installed into SMF. SMF includes the following bundles:

```
cm
deviceaccess
driverreaper
fileadmin
httpservice
logservice
prefs
servlet
smfadmin
tracker
useradmin
```

bundlesdata
bundlesdata is the default directory where the bundle natives and the bundle data files are stored when the Flash platform is used.

© Copyright IBM Corp. 1999, 2002

docs/ The Service Management Framework Guide document includes information about where to find the information you need to develop applications and how to develop and deploy bundles. The `docs/` directory contains SMF and OSGi documentation.

flashplatform.jar
flashplatform.jar contains the class for the flash platform.

jarbundles/
This folder contains an SMF installation using JAR format bundles. This installed bundle set can be used by SMF when running under both WebSphere Studio Device Developer 5.0 and Java Development Kit (JDK). This installation is used by the `smfjdk` command.

jxebundles/
This folder contains an SMF installation using J9 Executable (JXE) format bundles. This installed bundle set can be used by SMF only when running under WebSphere Studio Device Developer 5.0. See the WebSphere Studio Device Developer 5.0 documentation for more information on JXEs. This installation is used by the `smf` command.

makejxes
makejxes is a shell command that rebuilds JXE files for SMF Toolkit. The smf.jxeLinkOptions file is used to make SMF jxes. The bundle.jxeLinkOptions file is used to make bundle jxes.

makejxes.bat
makejxes.bat is a Windows command that rebuilds JXE files for SMF Toolkit. The smf.jxeLinkOptions file is used to make SMF jxes. The bundle.jxeLinkOptions file is used to make bundle jxes.

osgi.jar
osgi.jar contains Application Programming Interfaces (APIs) for the OSGi standards used in SMF, allowing you to compile your code against the APIs so that the bundles can run on any device that follows the OSGi standard. **osgi.jar** must be available in the class path when compiling, but must not be in the class path when running SMF.

resmanconsole.jar
resmanconsole.jar provides commands to create, analyze, and destroy memory spaces independent from the SMF Resource Manager. It requires the J9 JCL RM class library.

resmanimpl.jar
resmanimpl.jar provides the SMF Resource Manager that controls the resource conumption by bundles (memory, threads, sockets, and files). It requires the J9 JCL RM class library.

samples/
The **samples** folder includes sample bundles that demonstrate how to write bundles that run on the Framework. The samples included with SMF are:

```
Handle Security
Pizza
Simple Permission Policy
```

servlet.jar
The **servlet.jar** file defines the Java Servlet 2.1 APIs. The *servlet.jar* file is needed in the class path when compiling bundles that contain servlets, and

must not be in the class path when running SMF. The servlet.jar file is also a bundle which is installed in SMF to provide the servlet API to other bundles.

smf
smf is a shell command that starts SMF using the WebSphere Studio Device Developer 5.0 runtime.

smf.bat
smf.bat is a Windows command that starts SMF using the WebSphere Studio Device Developer 5.0 runtime.

smf.jar
smf.jar is the Service Management Framework and contains the class files for SMF. This file is used by the `smfjdk` command.

smf.policy
smf.policy grants smf.jar AllPermissions so the Framework can run in a secure environment.

smf.properties
smf.properties is the file that contains properties for SMF. You can edit this file.

smfbd.jar
Contains the classes needed by SMF to interact with the SMF Bundle Server, including snapshot support. It also contains the IDEAgent, which enables SMF to communicate with the IDE.

smfconsole.jar
Contains all classes needed to launch SMF with a command line console.

smfjdk
smfjdk is shell command that starts SMF using the JDK runtime.

smfjdk.bat
smfjdk.bat is a Windows command that starts SMF using the JDK runtime.

smftoolkit.readme
The **smftoolkit.readme** file includes last minute information about SMF.

smf-foundation.jxeLinkOptions
Contains the smart linker options to create the `smf-foundation-be.jxe` and `smf-foundation.jre files`.

smfres.properties
smfres.properties is the file that contains the properties for the SMF Resource Manager. You can edit this file.

smf-foundation_be.jxe
smf-foundation_be.jxe is the big endian SMF Toolkit in the JXE format. ThisJXE contains the WebSphere Studio Device Developer 5.0 jclMax Java class libraries.

smf-foundation_le.jxe
smf-foundation_le.jxe is the little endian SMF Toolkit in the JXE format. This JXE contains the WebSphere Studio Device Developer 5.0 jclMax Java class libraries.

smf-rm.jxeLinkOptions
smf-rm.jxeLinkOptions contains the smartlinker options to create the smf-rm_be.jxe and smf-rm_le.jxe files.

smf-rm_be.jxe
smf-rm_be.jxe is the big endian SMF Toolkit that contains pre-loaded Java

classes and resources in the JXE format. This file is used to run SMF on big-endian platforms (such as PowerPC® or ARM) using the WebSphere Studio Device Developer 5.0 runtime. This JXE contains the WebSphere Studio Device Developer 5.0 jclGwp Java class libraries.

smf-rm_le.jxe

smf-rm_le.jxe is the little endian SMF Toolkit that contains pre-loaded Java classes and resources in the JXE format. This file is used to run SMF on little-endian platforms (such as x86) using the WebSphere Studio Device Developer 5.0 runtime. This JXE contains the WebSphere Studio Device Developer 5.0 jclGateway Java class libraries.

Notices

This information was developed for products and services offered in the U.S.A. IBM might 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 might have patents or pending patent applications covering 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 information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 1999, 2002

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written.

These examples have not been thoroughly tested under all conditions.

No warranty

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CAN NOT BE EXCLUDED, IBM MAKES NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE WARRANTY OF NON-INFRINGEMENT AND THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY. IBM MAKES NO WARRANTY REGARDING THE CAPABILITY OF THE PROGRAM TO CORRECTLY PROCESS, PROVIDE AND/OR RECEIVE DATE DATA WITHIN AND BETWEEN THE 20TH AND 21ST CENTURIES. The exclusion also applies to any of IBM's subcontractors, suppliers, or program developers (collectively called "Suppliers").

Limitation of Liability

NEITHER IBM NOR ITS SUPPLIERS WILL BE LIABLE FOR ANY DIRECT OR INDIRECT DAMAGES, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST SAVINGS, OR ANY INCIDENTAL, SPECIAL, OR OTHER ECONOMIC CONSEQUENTIAL DAMAGES, EVEN IF IBM IS INFORMED OF THEIR POSSIBILITY. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE EXCLUSION OR LIMITATION MAY NOT APPLY TO YOU.

You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *2002* All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

For the XML Parser: The Apache Software License, Version 1.1 Copyright (c) 1999-2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: ″This product includes software developed by the Apache Software Foundation (http://www.apache.org/).″

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names ″Xerces″ and ″Apache Software Foundation″ must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.5. Products derived from this software may not be called ″Apache″, nor may ″Apache″ appear in their name, without prior written permission of the Apache Software foundation.

THIS SOFTWARE IS PROVIDED ″AS IS″ AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., http://www.apache.org. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Trademarks

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, or other countries, or both:

IBM, PowerPC DB2, WebSphere, and VisualAge®

Other company, product or service names may be trademarks or service marks of others.

OSGi™ is a trademark of the Open Service Gateway Initiative.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corp. in the U.S., or other countries or both.

Java and all Java-based marks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Other company, product and service names may be trademarks or service marks of others.