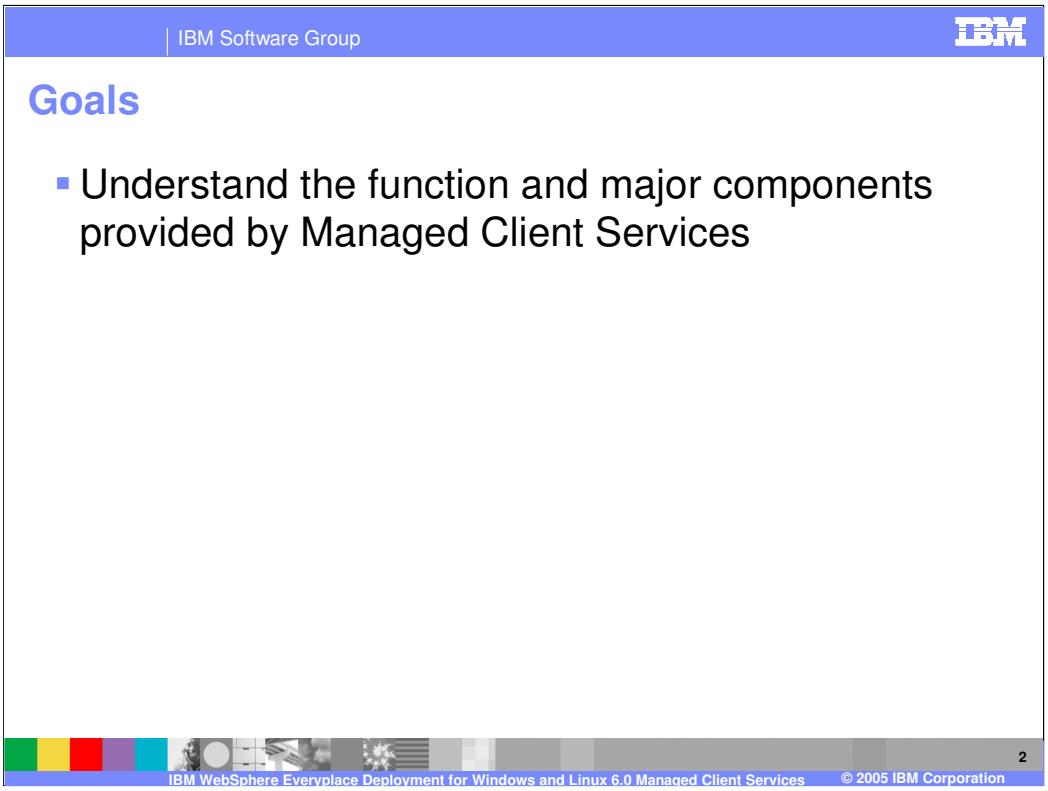
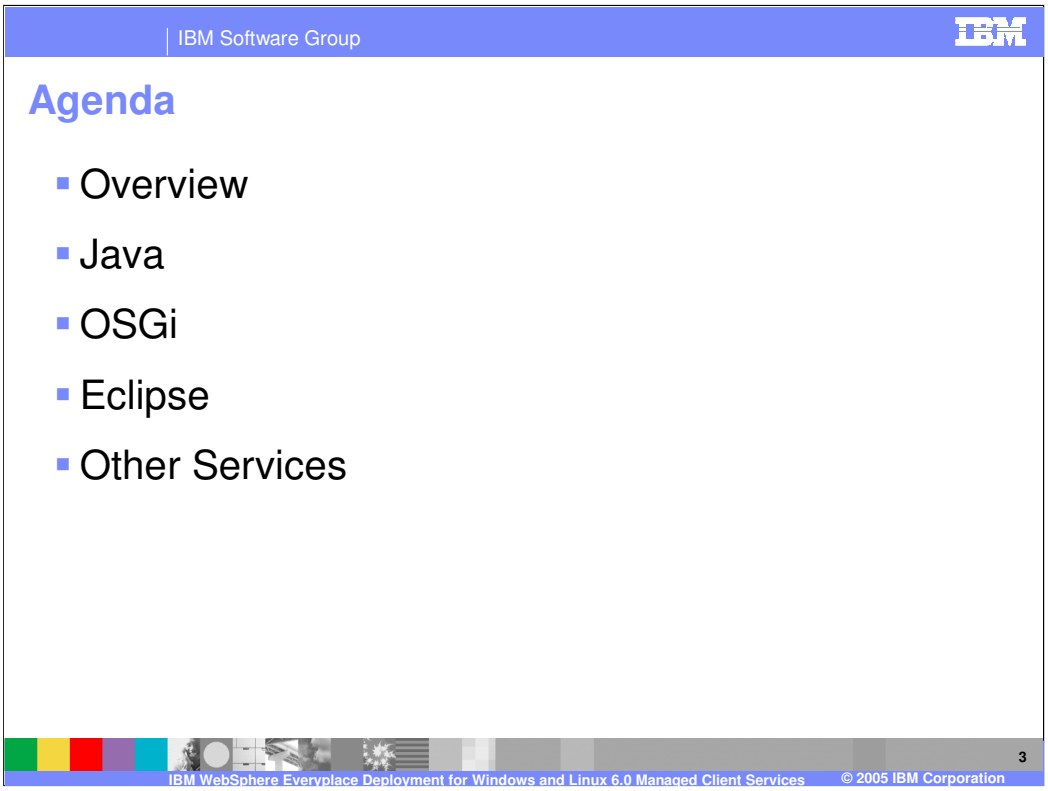
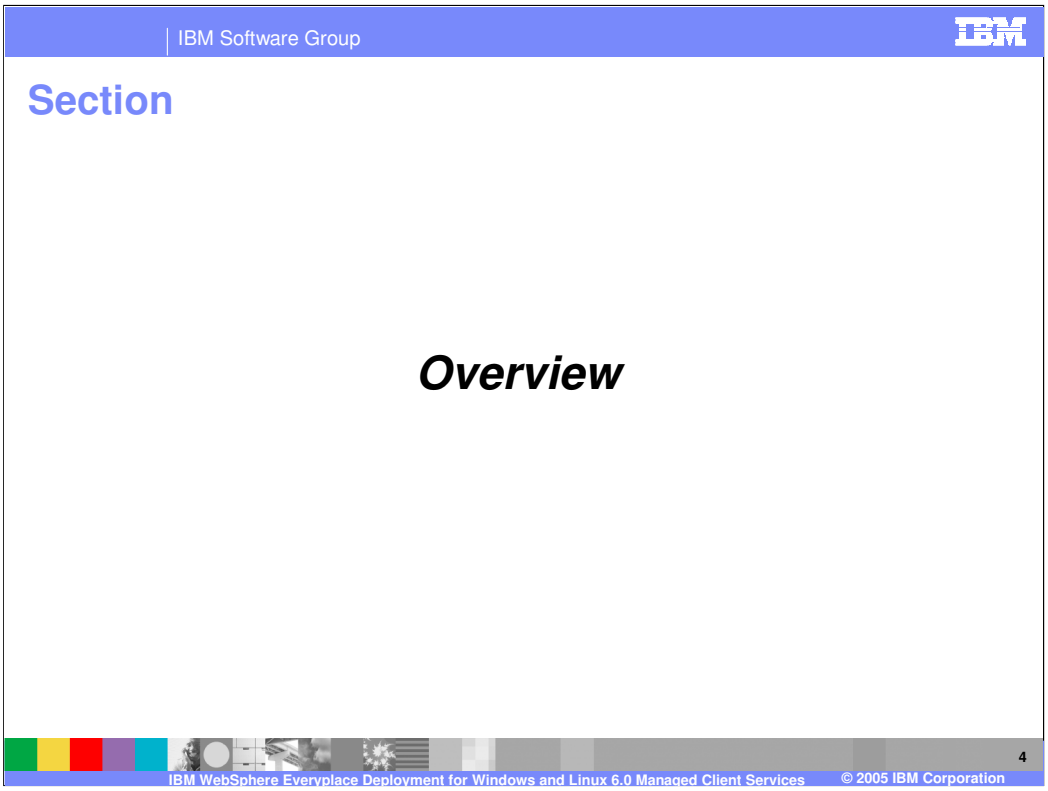
This presentation explains the Managed Client Services provided by IBM WebSphere Everyplace Deployment for Windows and Linux Version 6.



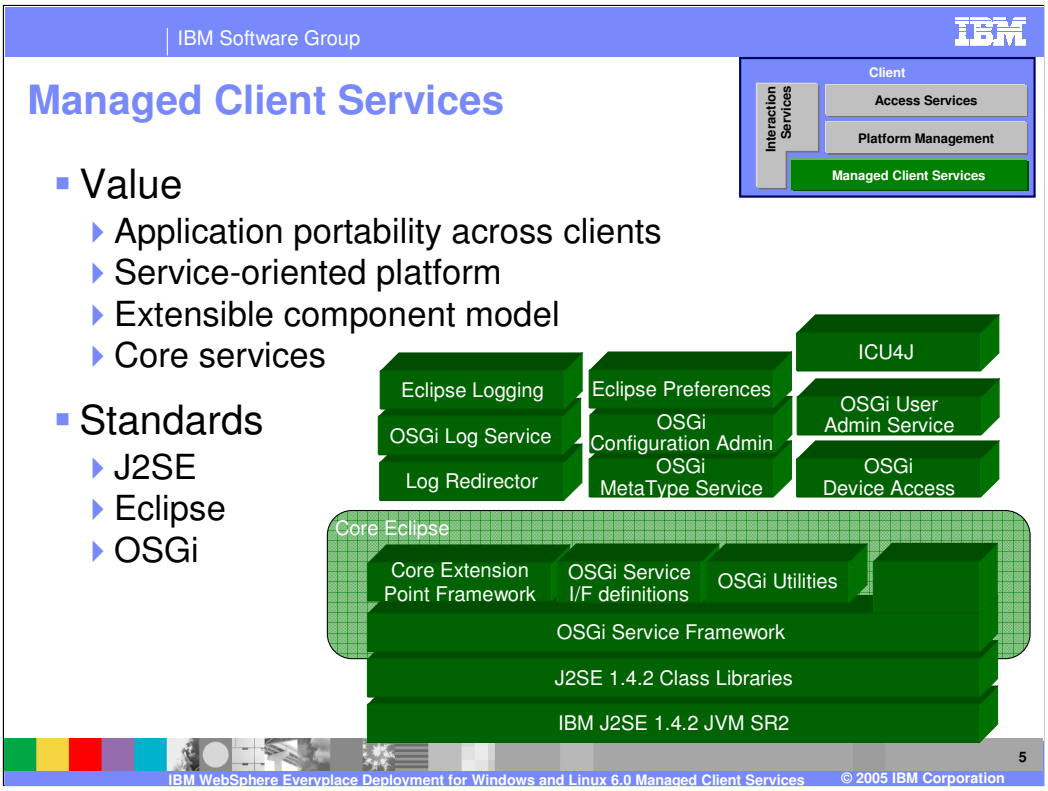
The goal of this presentation is to explain the Managed Client Services provided by IBM WebSphere Everyplace Deployment for Windows and Linux Version 6.



The agenda of this presentation is to provide an overview of managed client services, describe the Java Runtime Environment, explain the OSGi framework, cover the key components of Eclipse, and describe other services.



Let’s start with an overview of the Managed Client Services.



The client platform ships IBM's Java 2 Standard Edition (J2SE) 1.4.2 JVM (service release 2) as the base Java runtime environment.

The client platform provides an OSGi Service Framework that implements the OSGi R3 framework specification and provides a service-oriented architecture on top of the JVM. The OSGi Service Framework is the foundation of Eclipse and the client platform. OSGi Services provide the interface definitions for standard services defined by the OSGi R3 specification and OSGi Utilities provides an implementation of the OSGi utilities interface. The Eclipse Core Extension Point Framework supports the Eclipse plug-in component model and, by running with the OSGi Service Framework, enables plug-ins to receive the corresponding benefits of OSGi. All of these components ship in the core Eclipse framework.

The client platform provides implementations of additional services to assist you in the development of your applications including logging, preferences, globalization, and more.

Next, let's understand the Java Runtime Environment that ships with the client platform.

Java Virtual Machine

J2SE 1.4.2 Class Libraries

IBM J2SE 1.4.2 JVM SR2

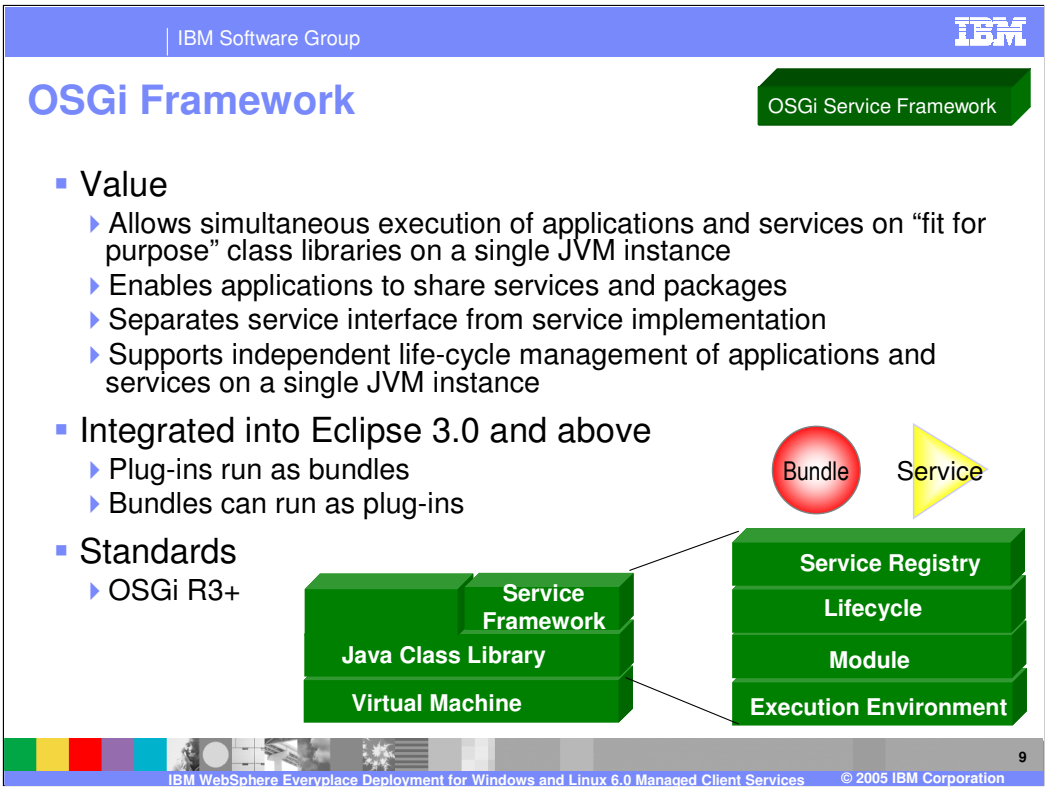
- IBM Java Virtual Machine shipped with installation
- Customers must use the JVM provided
- Provided as a plug-in
- Uses the –Xj9 vm argument
- Windows XP

  java version "1.4.2"
  Java(TM) 2 Runtime Environment, Standard Edition (build 2.2)
  IBM J9SE VM (build 2.2, J2RE 1.4.2 IBM J9 2.2 Windows XP x86-32 j9n142-20050609 (JIT enabled)
  J9VM - 20050524_1738_lHdSMR
  JIT - r7_level20050518_1803)

- Linux

  java version "1.4.2"
  Java(TM) 2 Runtime Environment, Standard Edition (build 2.2)
  IBM J9SE VM (build 2.2, J2RE 1.4.2 IBM J9 2.2 Linux x86-32 j9xia32142-20050609 (JIT enabled)
  J9VM - 20050524_1738_lHdSMR
  JIT - r7_level20050518_1803)

Next, let's explore the OSGi service framework, which is a critical component of the client platform.



The OSGi framework specification is provided by the OSGi Alliance. The OSGi Alliance's mission is to specify, create, advance, and promote wide industry adoption of an open service delivery and management platform. Incorporating the OSGi standard into the client platform provides four very important capabilities:

1. It enables multiple applications and components to share a single Java Virtual Machine (JVM). This saves valuable resources on the client when running multiple applications because only one instance of the JVM is launched rather than multiple instances of the JVM.
2. It enables applications to share services and packages, which further reduces resource requirements on devices.
3. It separates service interface from service implementation and provides publish, find, and bind operations in support of a service-oriented architecture. This capability enables integration of business applications on the same device.
4. It enables dynamic life-cycle management without a VM restart so components can be updated without impacting other unrelated components that are running at the same time.

The Eclipse framework is built on the OSGi Service Framework, which provides the Eclipse with powerful capabilities, such as the ability to dynamically load and unload plug-ins without restarting the Eclipse framework and robust life cycle management of plug-ins. Therefore, you can define each component in your applications as a plug-in, a bundle, or both depending on your requirements.

OSGi Framework

OSGi Service Framework

- OSGi Framework supplied by Eclipse implementation
- Meets the OSGi Release 3 specification
- No optional services are supplied by the Eclipse implementation
  - e.g. Configuration Admin Service, Log Service, Preferences Service
- Extensions to the framework added for Eclipse usage – not part of OSGI Release 3
  - e.g. Extension Points, Eclipse-AutoStart attributes, Require-Bundle/Provide-Package support

Bundle

OSGi Service Framework

- Packaging of a Java library/service/application for deployment into an OSGi framework
  - Dynamic life-cycle
  - Life-cycle event notification
  - Supports any resources (Java, Native, Images, etc.)
  - OSGi-specific manifest headers
- Bundles are made locally resident
- Plug-ins are Bundles

MANIFEST.MF

OSGi Service Framework

- Plug-in identity (Bundle-SymbolicName)
- Classpath (Bundle-Classpath)
- Startup class (Bundle-Activator)
- Dependencies

```
Bundle-Name: Sample
Bundle-SymbolicName: com.ibm.t3.sample
Bundle-Activator: com.ibm.foo.bar.impl.Activator
Bundle-Version: 1.5.7
Bundle-Classpath: foo.jar
Import-Package: org.osgi.service.log
Export-Package: com.ibm.foo.bar
Provide-Package: com.ibm.foo.bar
Import-Service: com.ibm.osg.service.log.LogService
Export-Service: com.ibm.foo.bar.SomeService
Require-Bundle: org.eclipse.osgi.services
```

Defines packages required by this bundle

Defines packages exported by this bundle

Defines packages provided by this bundle

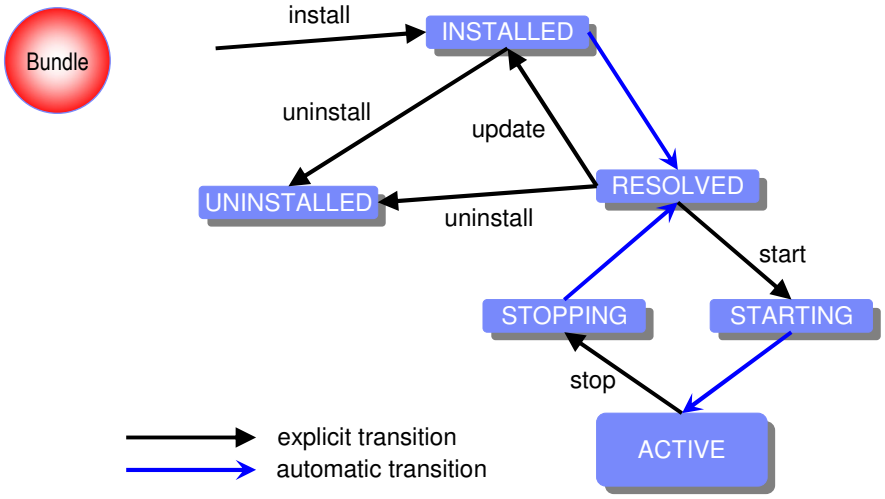
Defines services required by this bundle

Defines services exported by this bundle

Defines bundles required by this bundle

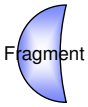
Bundle Lifecycle

OSGi Service Framework



Fragments

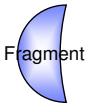
OSGi Service Framework



- Extends bundles to add capability
- Associates with one and only one host bundle
- Contains MANIFEST.MF file
- No Bundle-Activator
- Follows life cycle of host bundle
- Host bundle may have multiple fragments
- Typical uses
  - National Language Versions
  - Operating System specific implementations

Fragment MANIFEST.MF

OSGi Service Framework



- Fragment Identity (Bundle-SymbolicName)
- Host plug-in identity (Fragment-Host)
- Host classpath additions (Bundle-Classpath)
- Dependencies (same as bundle)

```
Manifest-Version: 1.0
Bundle-Name: HTTP Service for Web Container
Bundle-SymbolicName: com.ibm.pvc.webhttpservice
Bundle-Version: 1.0.0.20050908
Bundle-ClassPath: webhttpservice.jar
Bundle-Vendor: IBM
Fragment-Host: com.ibm.osg.service.http;bundle-version="[2.1.3,2.1.4)"
Import-Package: com.ibm.pvc.webcontainer.listeners; specification-
version=1.0,
 org.osgi.service.webapplication
Import-Service: com.ibm.osg.webcontainer.WebContainer
Export-Service: org.osgi.service.http.HttpService
```

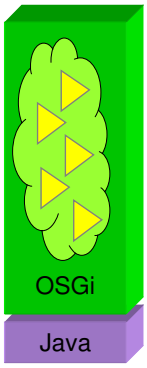
Services

OSGi Service Framework

- Java object
  - Registered by a bundle
  - Used by other bundles
- Specified by a Java interface
  - Clean separation of service specification and service implementation
  - Different bundles can register different implementations of the same service

Service Registry

OSGi Service Framework

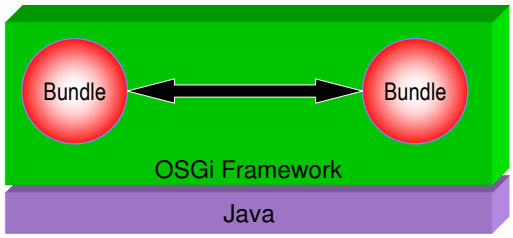


- Dynamic registry of available services
  - Secured by permissions
- Service object registered
  - Under one or more interface names
  - With optional properties
- Lookup mechanism
- Service Factory
  - Callback to create custom service objects
- Life cycle event notification
  - Registered, Modified, Unregistering

Collaboration

OSGi Service Framework

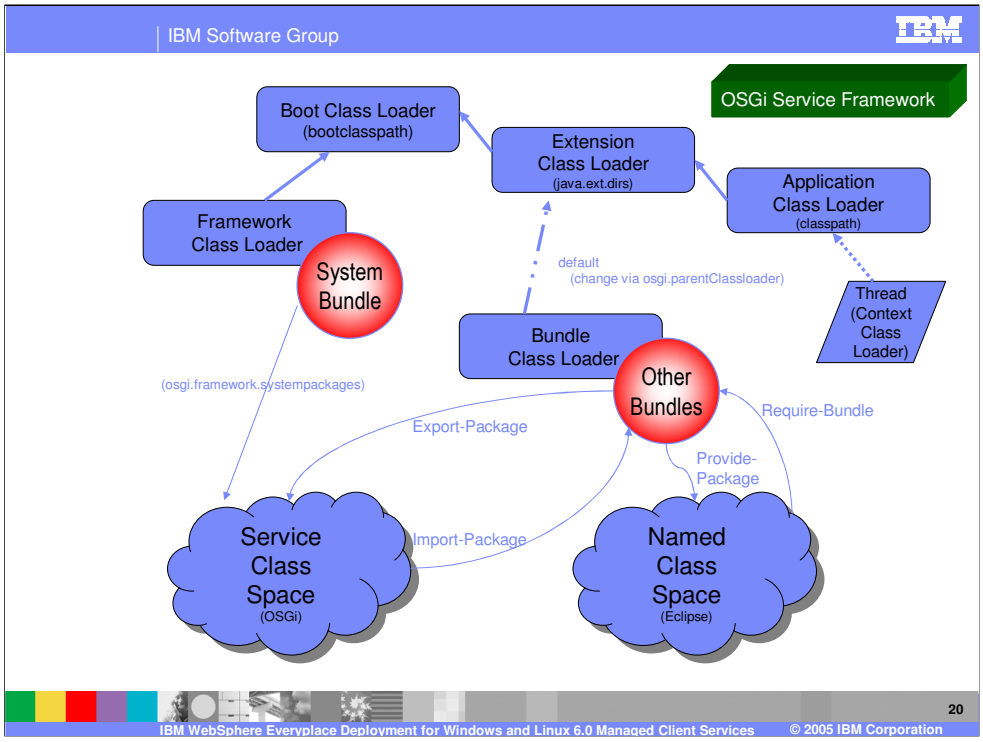
- Bundles can collaborate
  - Import/Export Services
  - Import/Export Packages
- Dependencies managed



Package Sharing

OSGi Service Framework

- Each bundle has its own ClassLoader
  - Bundles have separate namespaces
- Java classes are grouped into packages
  - Bundles can offer to share a package – export
  - Bundles can request to use a package – import
- Global shared package namespace
  - Class Loaders connected to allow classes to be shared
- Versioning
  - Only one version of a package may be shared at a time - defined by specification version



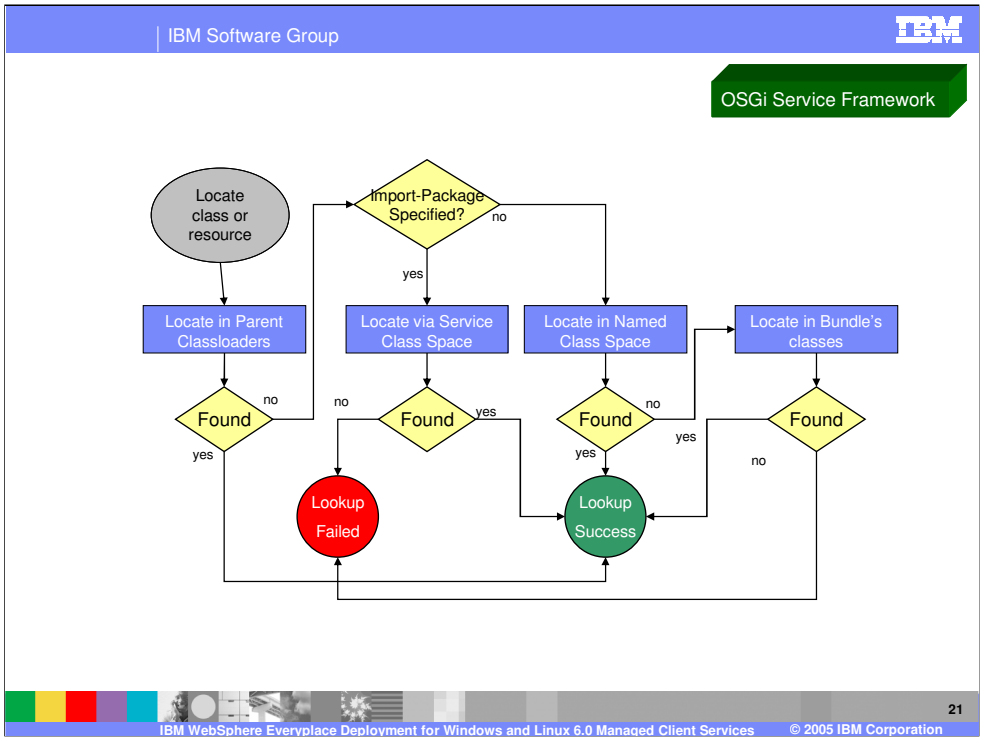
Class loading functions differently than a typical Java application because the client platform is built on the OSGi Service Framework. Since the mechanics for supporting plug-ins are implemented by using the OSGi Service Framework, a plug-in is the same as an OSGi bundle for the purpose of this explanation. The bundle and its associated classes specify and implement the process for Java class loading, prerequisite management, and the bundle's life cycle.

Each bundle installed and resolved in the OSGi Service Framework must have a class loader. This class loader, called the Bundle Class Loader, provides each bundle with its own name space to avoid name conflicts and enables package sharing with other bundles.

The Bundle Class Loader searches for classes and resources in the bundle's class path as defined by the Bundle-Classpath header in the bundle's manifest. The Bundle Class Loader has a parent class loader as specified in the osgi.parentClassloader property. By default, the parent class loader is the Extension Class Loader for the client platform. However, the Extension Class Loader also has a parent class loader - the Boot Class Loader. As a result, the parent of the Bundle Class Loader actually consists of the Boot Class Loader and the Extension Class Loader.

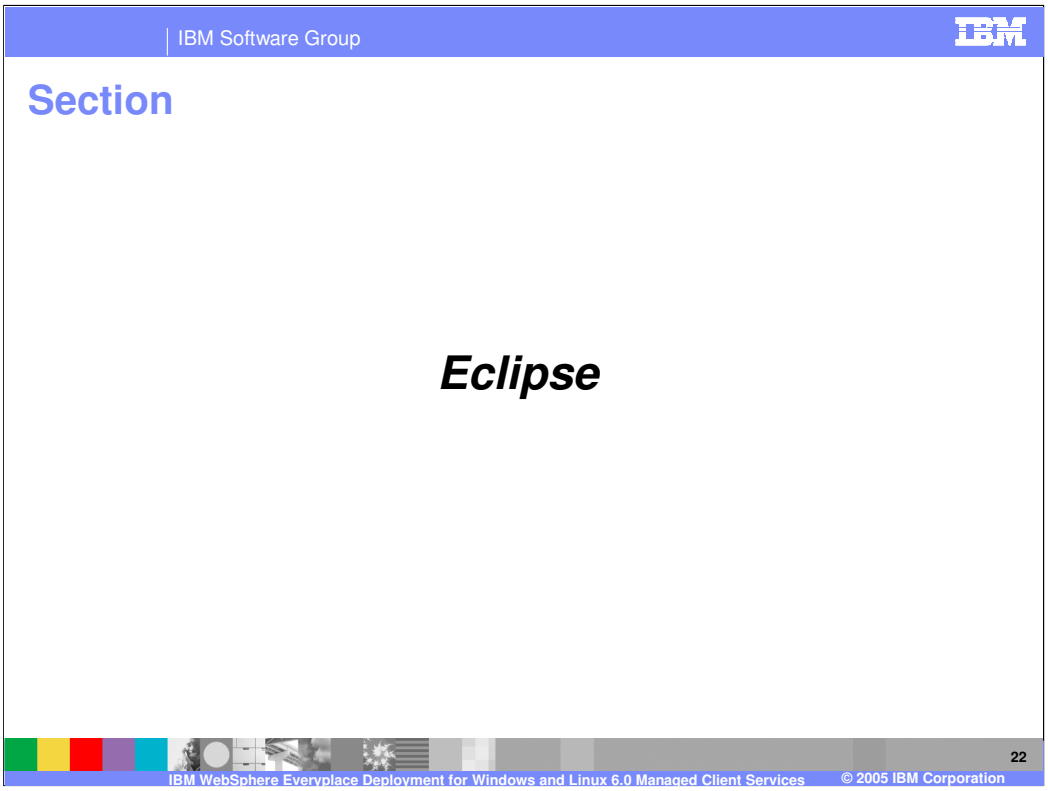
A bundle can export the classes and resources in one or more of its packages by specifying each such package name in the Export-Package header in its manifest. The classes and resources in each exported package become part of the Service Class Space and are made available to other bundles with permission to use the package A bundle can import one or more packages by specifying each package name in the Import-Package header in its manifest. If the bundle has permission to import these packages, then the bundle can use the classes and resources in these packages as defined in the Service Class Space. A package can be shared based on its name and, optionally, its version. However, if multiple bundles share (export) a package with the same name, then the OSGi Service Framework determines the bundle that shares that package with other bundles based on the highest version of the declared package. As a result, a bundle that imports a package must know the name of the package it needs to import but cannot explicitly control which bundle provides the package it actually uses.

A bundle can also provide the classes and resources in one or more of its packages by specifying each such package name in the Provide-Package header in its manifest. The classes and resources in each provided package become part of the Named Class Space and are made available to other bundles with the appropriate permissions. A bundle can explicitly use packages provided by a bundle by specifying each required bundle in the Require-Bundle manifest in its header. The Require-Bundle manifest header contains a list of bundle symbolic names that need to be searched after the imports are searched but before the bundle's class path is searched. However, only packages that are marked as provided by the required bundles are visible to the requiring bundle.



To locate a class or resource, the search order is as follows:

1. The Bundle Class Loader delegates the request to its parent class loader (PARENT), which results in the Boot Class Loader and then the Extension Class Loader attempting to locate the class or resource. If the class or resource was found, then the class loader returns this result. If the class or resource was not found, then the search continues with the next step.
2. If the Bundle Class Loader determines that the requested class or resource is in a package imported from the Service Class Space (SERVICE) and it was found in the Service Class Space, then the class loader returns this result. If the class or resource is not found, then the request fails. If the Bundle Class Loader determines that the requested class or resource was not in the Service Class Space, then the search continues with the next step.
3. The Bundle Class Loader searches the Named Class Space (NAMED) and if the class or resource was found, then the class loader returns this result. Otherwise, the search continues with the next step.
4. The Bundle Class Loader searches its own internal class path and the internal class path of any attached fragment bundles. If the class or resource was found, then the class loader returns the results. If the class or resource was not found, then the search terminates and the request fails.



Next, let's review the Eclipse components in the client platform.

Eclipse

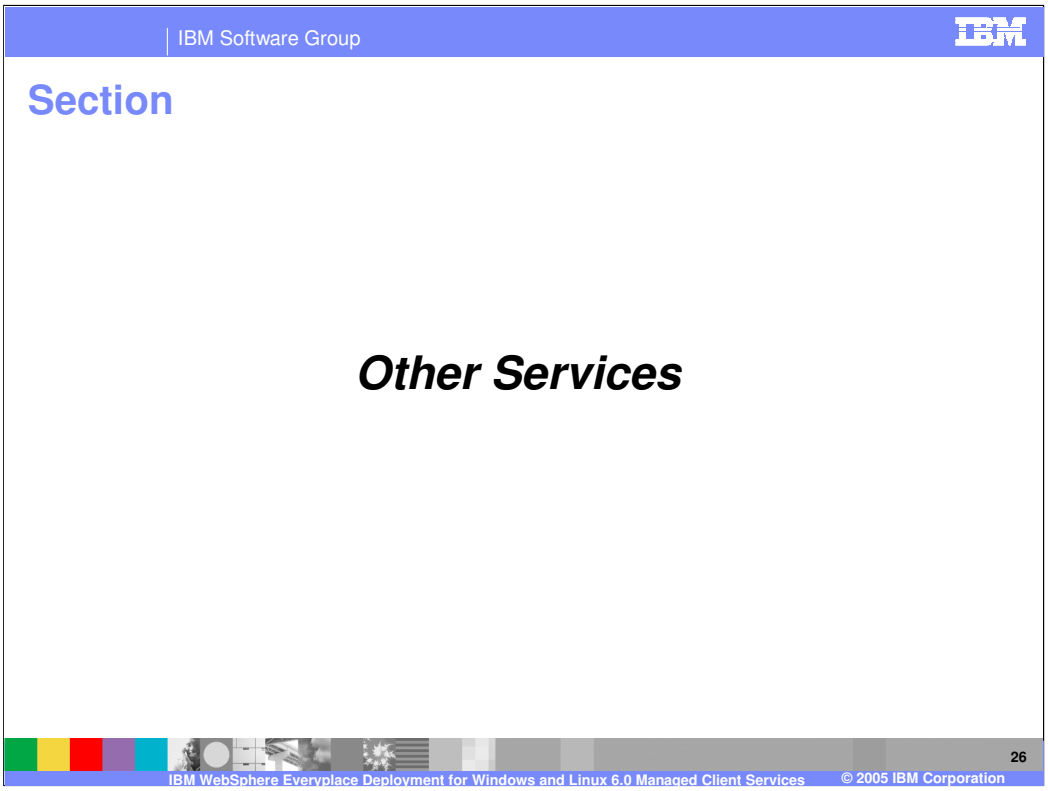
- IBM WebSphere Everyplace Deployment for Windows and Linux Version 6 includes the following Eclipse components:
  - Eclipse 3.0.2
  - Updates/patches to some Eclipse plug-ins
  - The Eclipse RCP (Rich Client Platform)
  - Additional plug-ins from the SDK
- Customers can add other Eclipse plug-ins not included by IBM
  - IBM does not support these plug-ins
  - Versions should be the Eclipse 3.0.2 version

plugin.xml

- Eclipse uses plugin.xml files
  - Extension points
  - Legacy
    - Plug-in identity
    - Classpath
    - Dependencies
    - Startup Class
- Legacy Information overlaps with OSGi MANIFEST.MF file contents
  - Framework always uses MANIFEST.MF
  - Legacy information in plugin.xml translated into MANIFEST.MF file
  - If MANIFEST.MF and plugin.xml contain conflicting information, MANIFEST.MF has precedence
- plugin.xml information cached when first processed

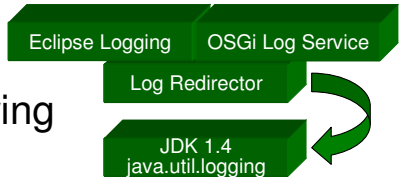
fragment.xml

- Eclipse uses fragment.xml files
  - Extension points
  - Legacy
    - Fragment identity
    - Host identify
    - Classpath
    - Dependencies
- Legacy Information overlaps with OSGi MANIFEST.MF file contents
  - Framework always uses manifests
  - Legacy information in fragment.xml translated into MANIFEST.MF file
  - If MANIFEST.MF and fragment.xml contain conflicting information, MANIFEST.MF has precedence



Finally, let's review the other managed client services available to you in the client platform.

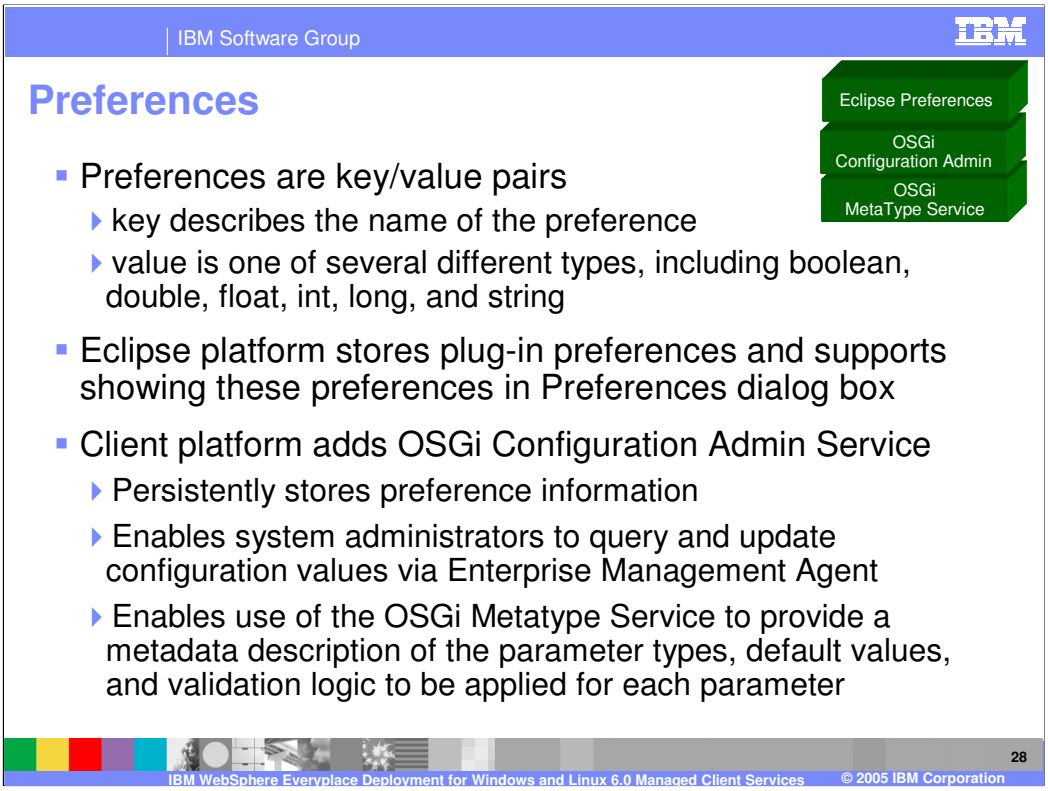
Logging Improvements



- Applications can use the following methods for logging messages
  - Eclipse logging interface
  - OSGi LogService interface
  - Standard error and standard out
  - java.util.logging interface (JDK 1.4)
- logRedirector – Collects log messages in one file
  - Captures messages logged via Eclipse logging interface, OSGi LogService, standard error and out
  - Maps logging levels to JDK 1.4 logging levels
  - Redirects messages to JDK 1.4 java.util.logging

You use logs to gather information about problems that might happen while using the client platform. The OSGi framework, Eclipse framework and the 1.4 JDK (JSR47) all provide different systems for logging messages. Applications in some cases also leverage standard error and standard out for message delivery, and these messages must also be captured to ensure all data is available during the problem determination stage. The client platform provides a plug-in, called the logRedirector, that collects all messages logged in the client platform into one persistent log file. The logRedirector captures messages logged from the OSGi logService, the Eclipse logging APIs, and standard error and standard out and redirects them to the JDK 1.4 java.util.logging. A java.utils.logging log file manager is also provided by the client platform, which supports configuration of the JDK logging and manages the persistent log file.

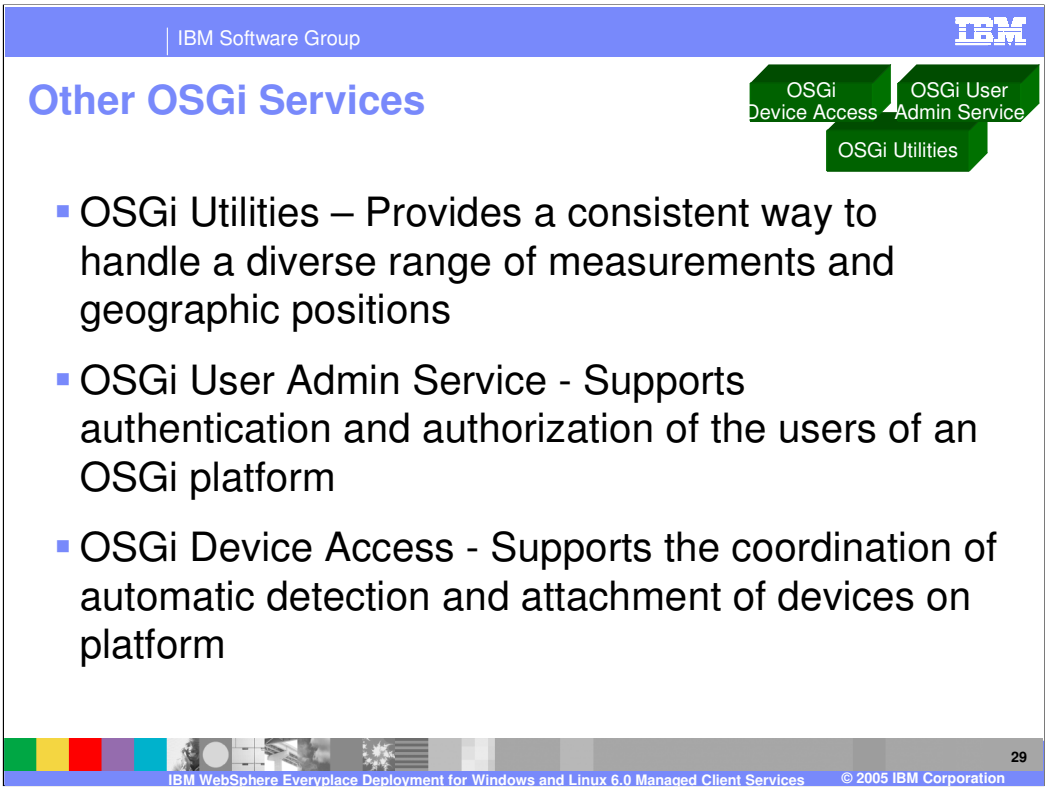
Each of the logging systems available in the client platform has its own definition of logging levels, which provide information on the severity and type of logging information. In order to bring all of the messages from these disparate systems together, the logging system maps logging levels from the Eclipse logging service and the OSGi LogService to the logging levels defined by JDK 1.4 (i.e. Severe, Warning, Info, Finest).



Preferences are key/value pairs, where the key describes the name of the preference, and the value is one of several different types, including boolean, double, float, int, long, and string.

The Eclipse platform provides support for storing plug-in preferences and showing them to the user on pages in the workbench Preferences dialog box.

The client platform extends the Eclipse capabilities by including the OSGi Configuration Admin service. Configuration Admin provides capabilities to store preference or configuration information based on key value pairs. Applications that use Configuration Admin will be notified when configuration information changes. Applications that use Configuration Admin to store configuration and preference information can also use the Metatype service to provide a metadata description of the information. The metadata can describe the parameter types, default values, and validation logic to be applied for each parameter. If Configuration Admin is used to store configuration information, system administrators can query and update configuration values via the Enterprise Management Agent.



Additional OSGi Services include:

•OSGi Utilities – Provides a consistent way to handle a diverse range of measurements and geographic positions

•OSGi User Admin Service - Supports authentication and authorization of the users of an OSGi platform

•OSGi Device Access - Supports the coordination of automatic detection and attachment of devices on platform

Additional Services

- International Components for Unicode for Java (ICU4J) – Supports globalization of applications
  - Support multiple locales
  - Support bidirectional text layouts
  - Create translatable components
- XML Parsers
  - Use standard Java API for XML Parsing (JAXP)
  - Use XML Parsing Service I/F to dynamically select parser at runtime – requires additional code

You can use International Components for Unicode for Java (ICU4J) to globalize your applications. ICU4J enables you to:

•Support multiple locales

•Support bidirectional text layouts

•Create translatable components

•The following Web site provides more information about the icu4j package: http://www-306.ibm.com/software/globalization/icu/index.jsp

You can also use XML Parsers to process XML data.

•Use the standard Java API for XML Parsing (JAXP)

•Use the XML Parsing OSGI Service I/F to dynamically select parser at runtime and be notified of parser service events by the XML Parser Service. However, using this interface requires additional code.

Trademarks, Copyrights, and Disclaimers

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

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. 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 in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.