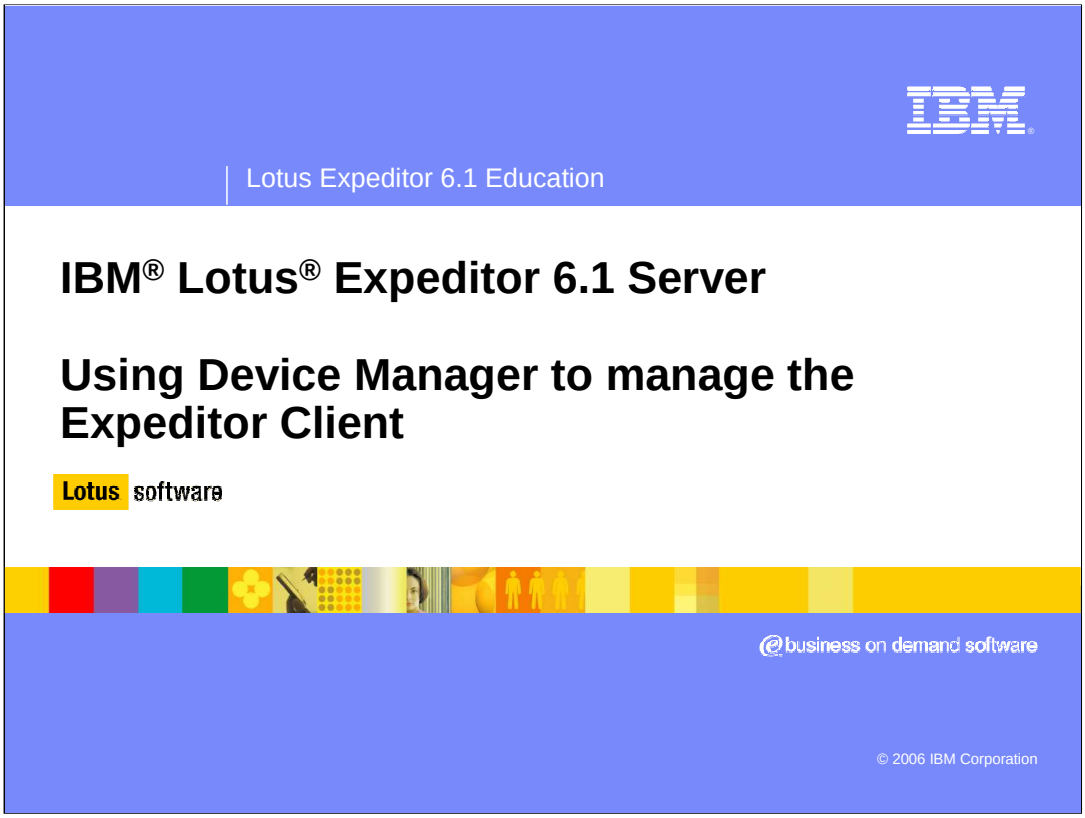
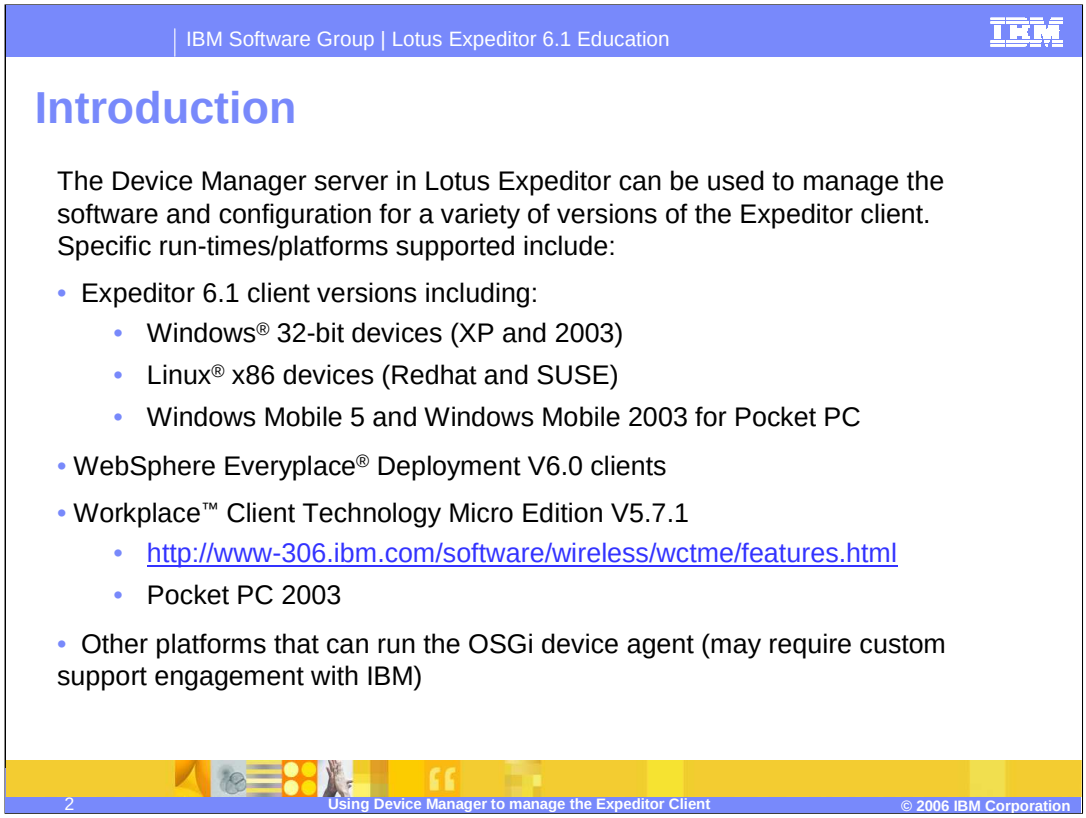


Hello, and welcome to this overview of using the Device Manager to manage the IBM Lotus Expeditor 6.1 Client.



The Lotus Expeditor 6.1 Device Manager, or DMS, provides a flexible solution for managing devices that is independent of connection type or device capabilities.

DMS provides device class support for Windows 32 desktops, Linux desktops, Windows Mobile 2003 devices, and client environments running the OSGi device agent, such as Workplace Client Technology Micro Edition 5.7.1 or 5.7.2.

Client management support

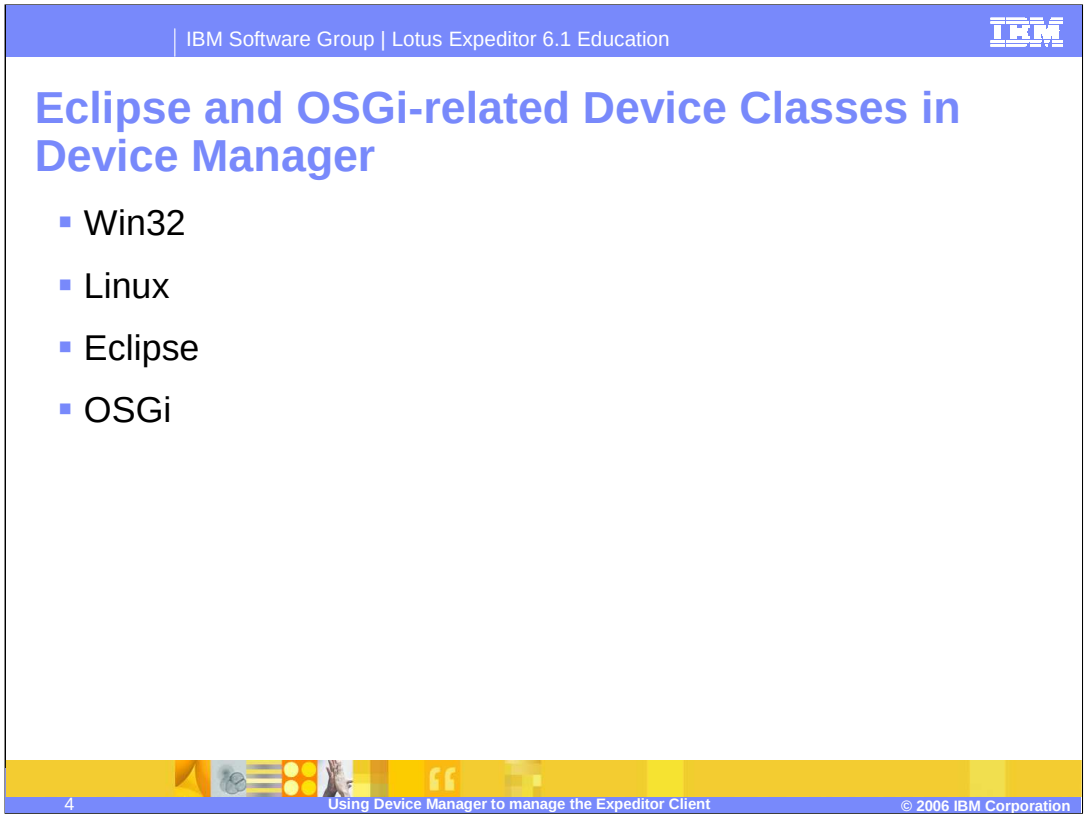
- Lotus Expeditor and WebSphere Everyplace Deployment client management
  - Configuration and Software Deployment
  - Native Software Installation
- Win32® and Linux extensions
  - Expanded inventory
  - Registry retrieval and editing

Management for the Lotus Expeditor clients is provided for the client run-time. Configuration and software deployment is provided within the run-time.

Limited capabilities for native software installation are provided, and intended for delivering content associated with custom applications such as images and data files.

Additional operating system platform management and inventory features are available in the Win32 and Linux derivatives of the Expeditor client.

Expanded inventory management is available for native hardware and software. Registry retrieval and editing is possible on Windows 32-bit platforms. These functions are intended for support of custom application deployment and configuration, not for general operating system or native application management.



Several device classes in Device Manager are related to management of the Lotus expeditor clients and share many common components and functions.

The specific device classes related to Expeditor client management include:

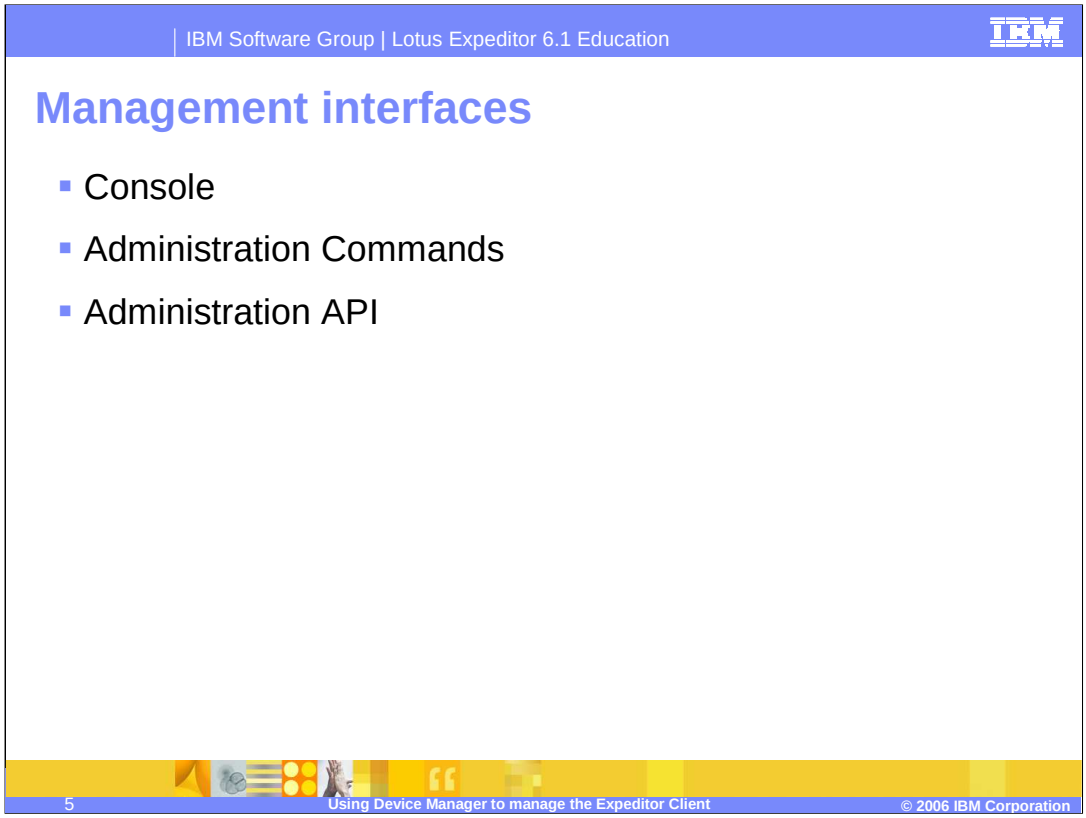
•Win32 for managing Windows 32 desktops,

•Linux for managing Linux desktops,

•Eclipse for managing Expeditor for devices,

•and OSGi for managing other client environments running the OSGi device agent.

The OSGi support is device neutral, which enables management of any platform capable of running the core OSGi device agent.



Device Manager provides three management interfaces for managing jobs, devices, servers, software, and queries.

•The Device Manager console provides a graphical user interface.

•The Device Manager Administration Commands provide a set of command line operations,

•and the Administration API provides a set of web **services** that can be used by other applications.

Device Manager provides several categories of jobs for managing the Expeditor clients. The categories are:

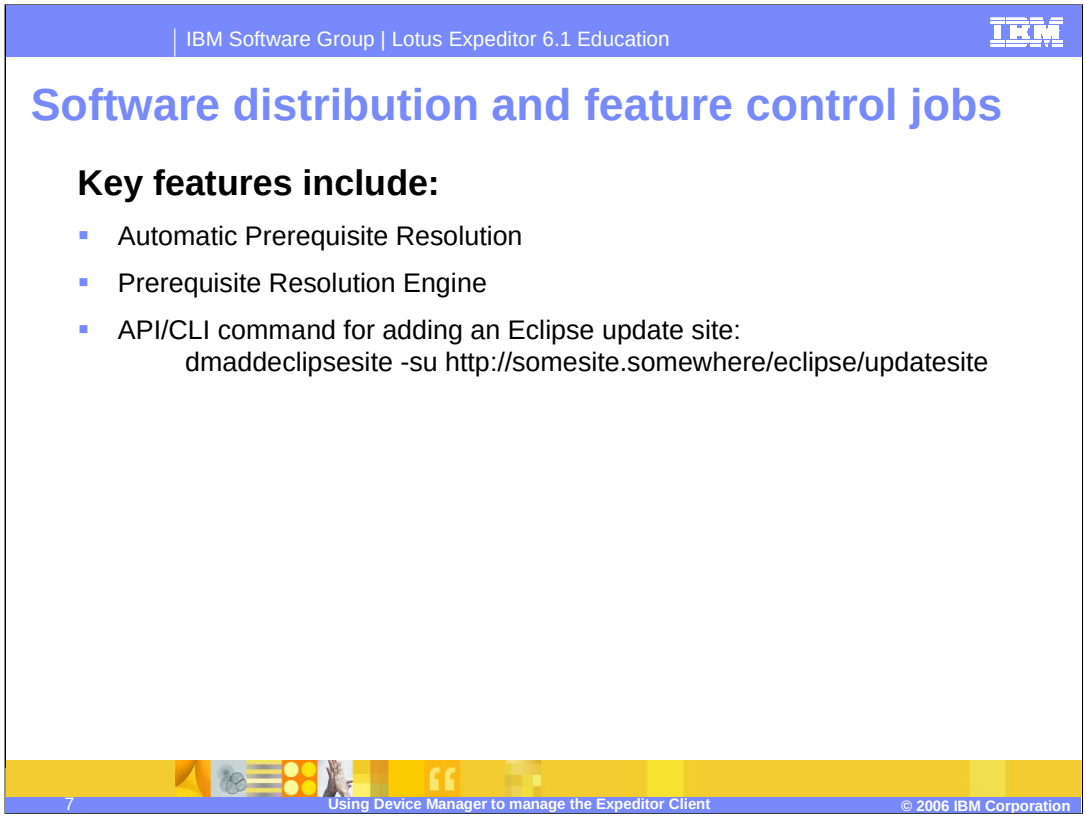
•high-level jobs such as software distribution and removal

•building block jobs, such as command script jobs

•Eclipse management which includes **jobs** specific to managing the Eclipse platform

•And, Configuration management, which includes jobs for retrieving and setting software configuration properties

We will go more in depth on the various device manager jobs available, on subsequent slides.



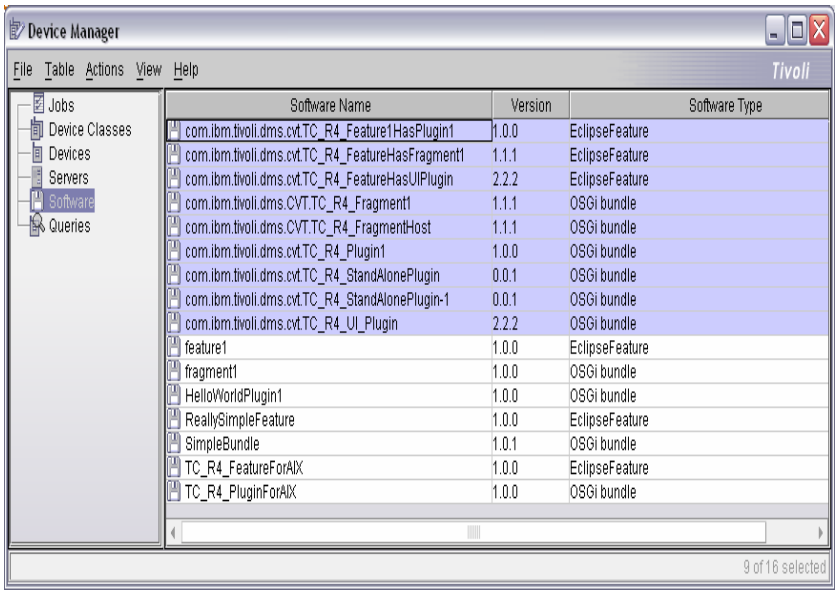
Software distribution and feature control jobs are used to install one or more software features or bundles into the client.

Automatic prerequisite resolution is used to resolve: Eclipse features, OSGi R4 bundles, services, packages and other resource requirements. Requested features or bundles and their prerequisites are distributed to the **device** only when they have not already been installed on the device.

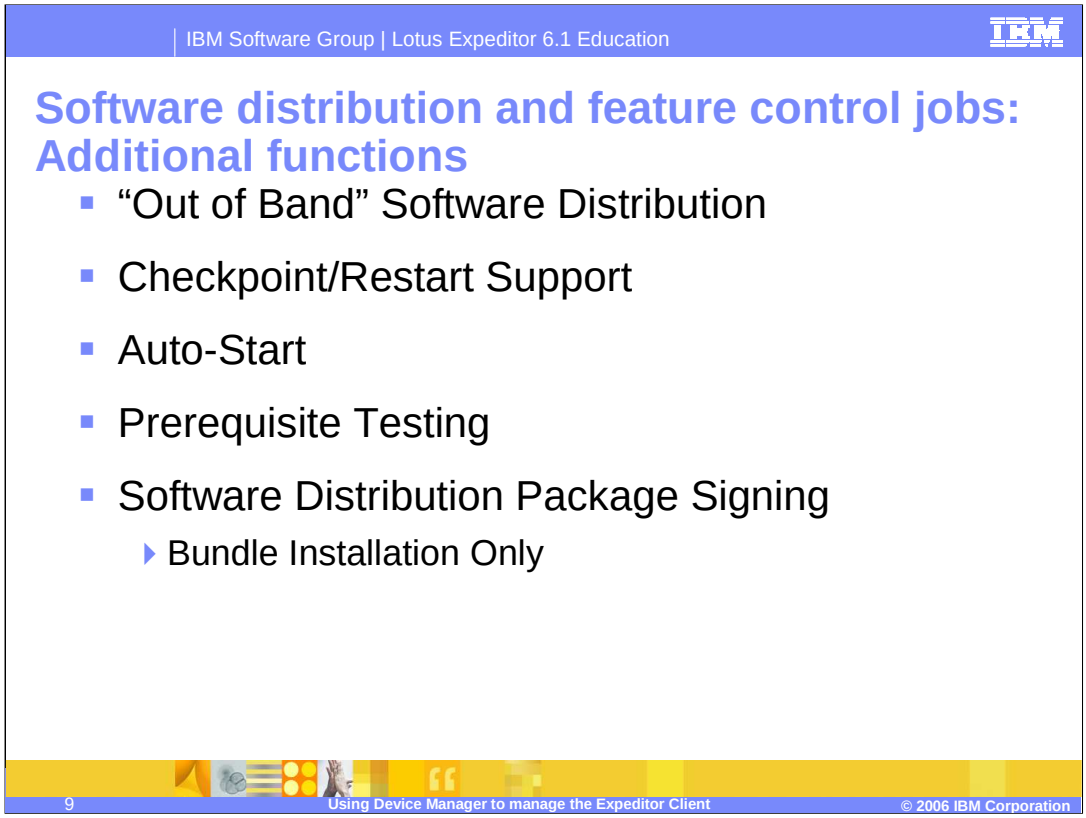
Prerequisite resolution also works in conjunction with the Eclipse Update Manager and Resolver engine to provide similar function utilizing a central management server. The engine allows for the delivery of prerequisites across multiple Eclipse update sites. If a prerequisite is missing, feedback is provided in the job progress. **Use** of the prerequisite engine is optional for Feature Control jobs.

Device Manager provides a **convenience API/CLI** command for registering a complete Eclipse update site. Running the **dmadd eclipse site** command automatically registers all features and plug-ins available at that site.

Example site registration



This slide shows an example of the Device Manager console after an Eclipse update site registration is completed. All features and plug-ins available at the site are automatically registered in the Device Manager database and are shown in the console software list.



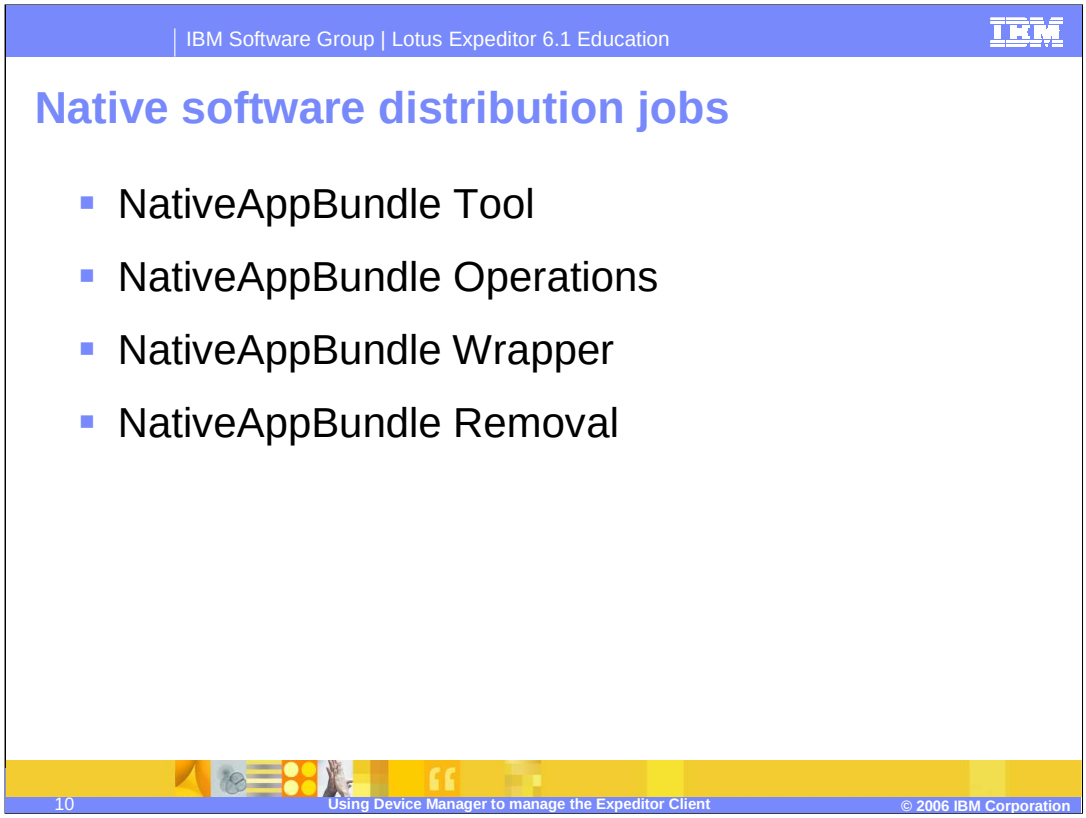
Device Manager uses “out of band” software distribution, which means all file transfers occur directly between the client and the package server using HTTP or HTTPS. This frees up the management server to perform other work.

If the transfer of files is interrupted by a loss of network connection, checkpoint and restart support ensures the file transfer is continued from where it left off once the connection is reestablished.

Bundle auto-start can be specified with the distribution job.

Feature control jobs provide a “verify only” option to test whether prerequisites can be satisfied at a single Eclipse update site. Device manager also provides a command, dmlssw, which can be used to test prerequisites against a repository or particular client.

Device Manager provides a security feature called Software Distribution Package Signing. This feature ensures the content of the OSGi bundles has not been modified between initial registration time and the actual delivery to the device. At software registration time, the Device Manager server saves the URL of the software package and a MD5 checksum of the software package in the database. At distribution time, the Device Manager server sends the URL and MD5 checksum to the device agent. The agent verifies the checksum against the actual package. If there’s a mismatch, the install is not attempted. This is not available for feature installations due to the use of the Eclipse Update Manager.



Native software distribution jobs are used to send “native” software content to the device.

The NativeAppBundle tool can be used to wrap a native application, such as file system content, in an OSGi bundle for subsequent distribution.

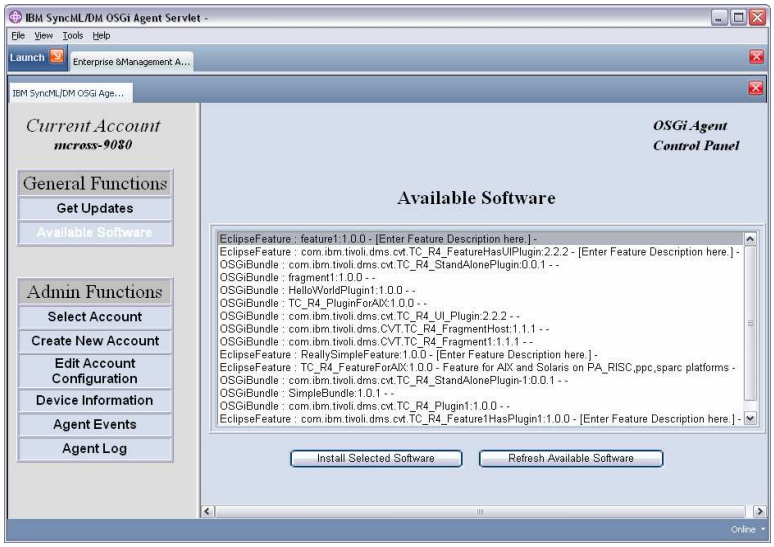
A native software distribution job can perform operations such as laying down files or directories on the local file system and launching installers or executables.

The NativeAppBundle wrapper makes the package appear as a normal OSGi bundle, but has special bundle activation and manifest definitions that are used at client run-time during installation in order to install native content. A small bundle without the native image remains resident on the OSGi system. This enables the deployed native applications to show up in the OSGi bundle inventory.

On native packages that have an appropriate uninstall command specified, the SoftwareRemoval and BundleControl jobs can be used to uninstall the native application.

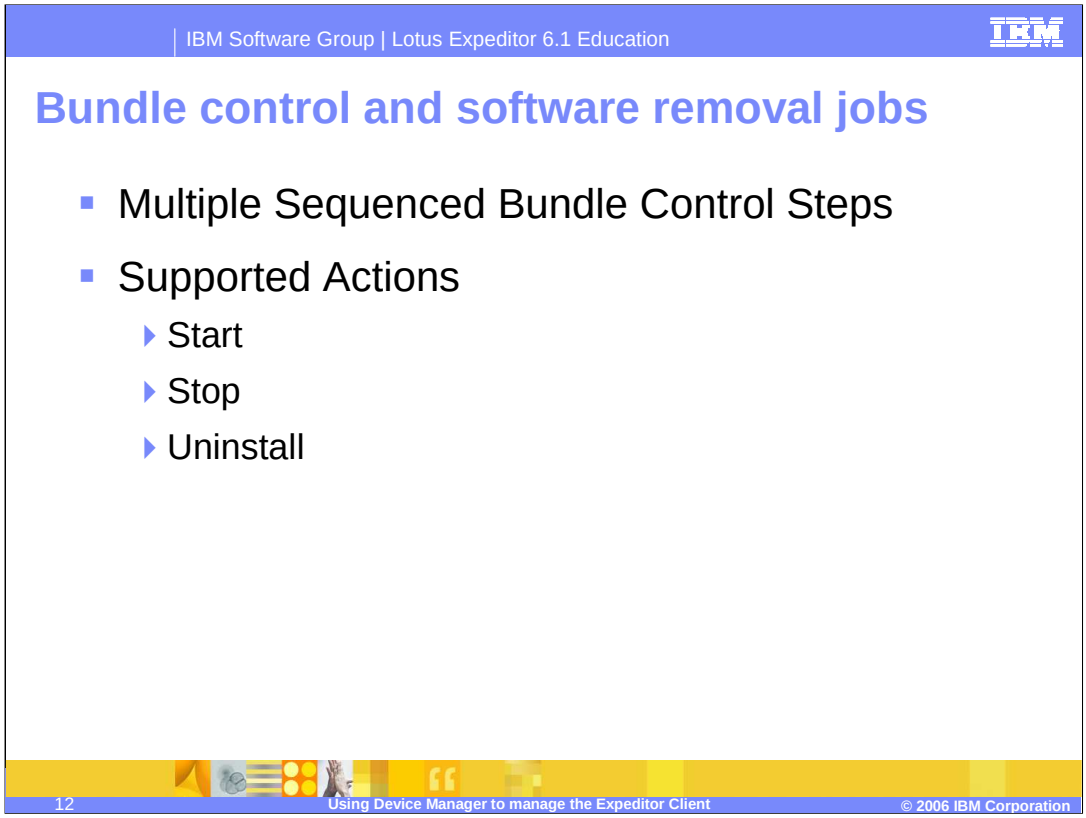
The native application bundle tool is shipped on the Expeditor Client media.

Software distribution - User pull sample user interface



An API is provided to enable applications running on the device agent run-time to get information from the server about what software is available for distribution. The API also enables the application to select one of these bundles to be distributed to the client, along with any required pre-requisites.

This functionality is demonstrated here by the standard GUI component of the device agent. This same functionality could be utilized by a custom User Interface for integration within another application.



Bundle control jobs are used to control the state and operation of bundles.

Multiple sequenced bundle control steps are supported.

Supported actions include starting, stopping and uninstalling a bundle.

The uninstall actions are also mapped to the Software Removal job type, as it shares an implementation with Bundle Control jobs.

Eclipse Inventory Collection jobs collect information about the set and state of installed features, preferences, and properties on the client.

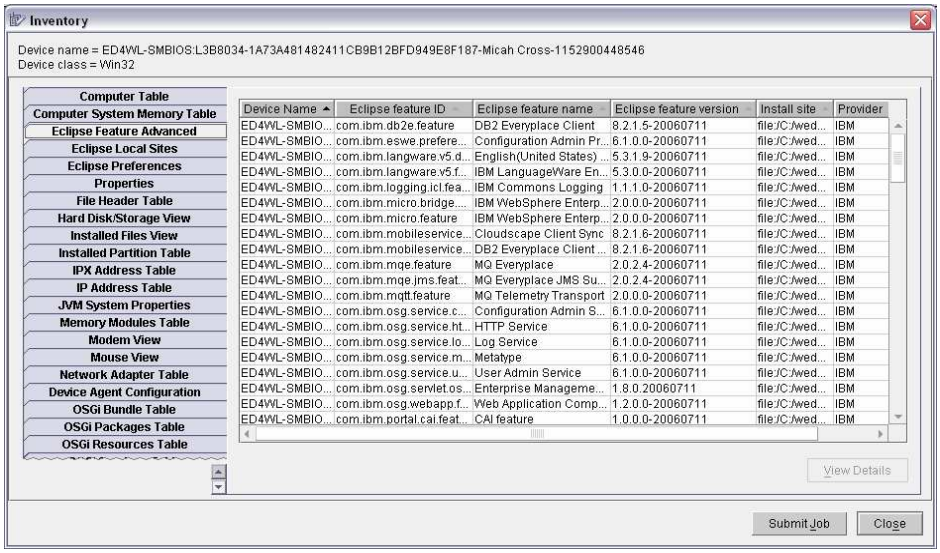
Eclipse feature information collected, includes: feature id, name, version, install site, provider and state.

Information about local Eclipse update sites can be collected, including site key and local installation path.

A Preferences Retrieval job can be used to obtain Eclipse preferences data, such as preference name, scope, type and value. These preferences can be modified using a Preferences Editing job.

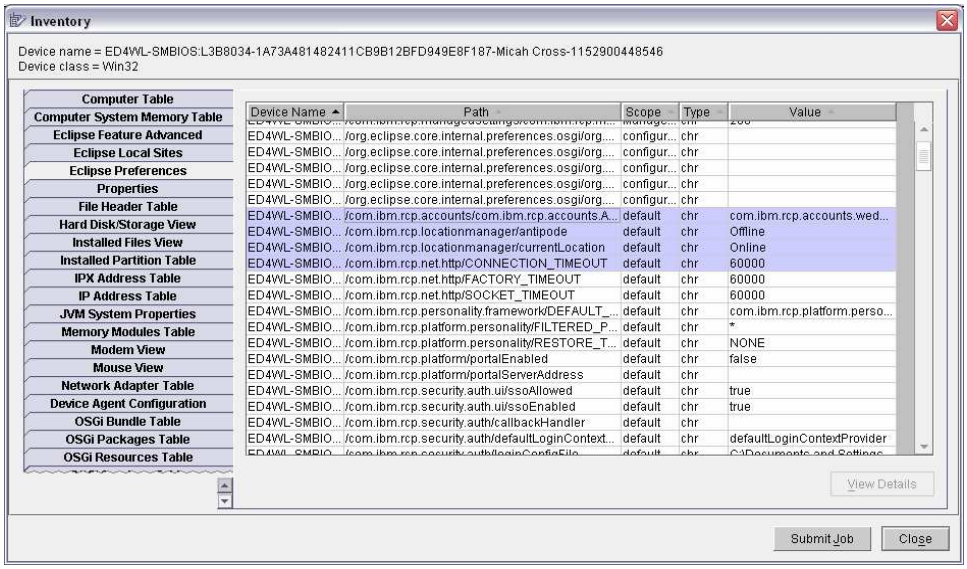
Client properties are obtained using a Properties Retrieval job. Property name, type and value can be viewed. These properties can be edited using a Properties Editing job.

Example Eclipse feature inventory



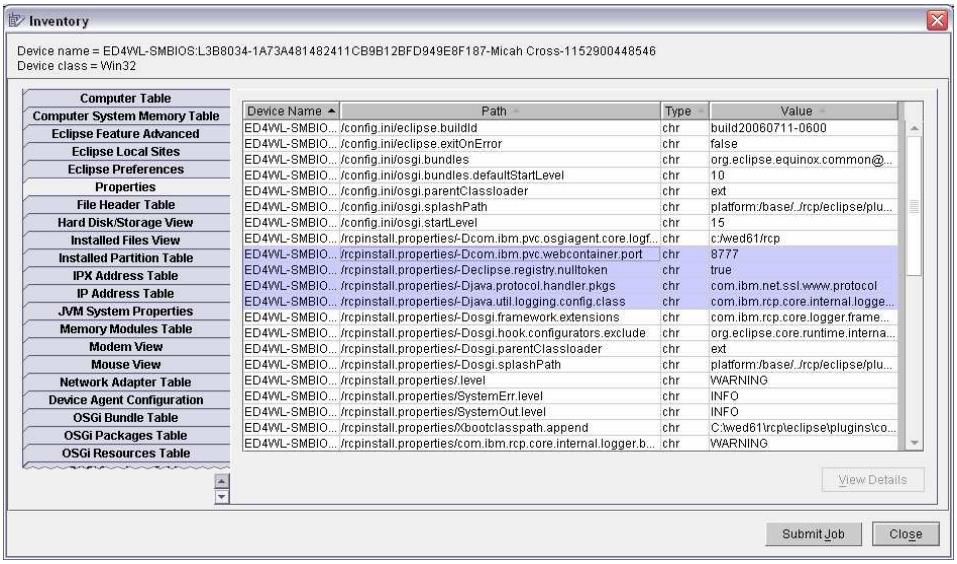
This slide shows an example of an Eclipse Feature Inventory. All Eclipse features for the device are listed along with identifying information.

Example Eclipse preferences inventory



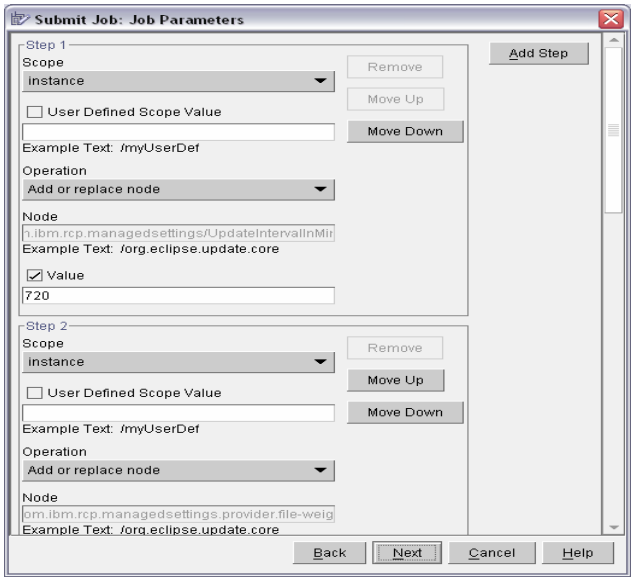
This slide shows the Eclipse Preferences Inventory screen. These are obtained from an Eclipse Preferences Retrieval Job.

Example client properties inventory

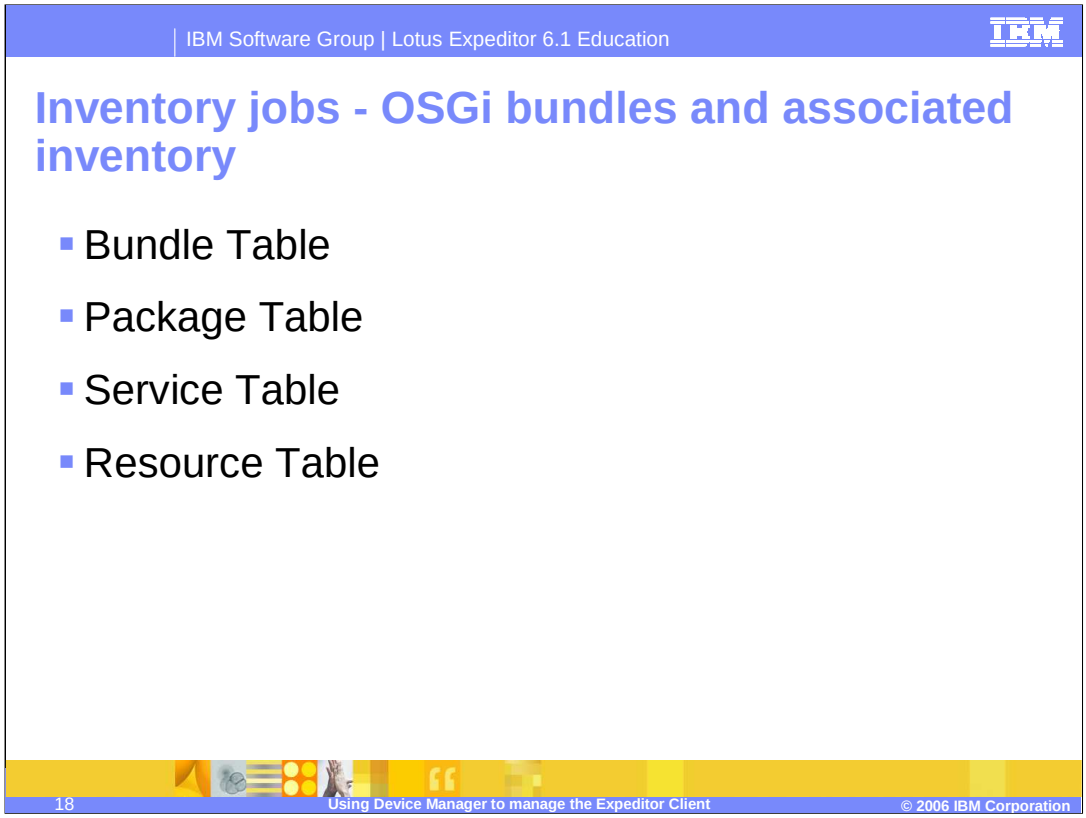


The client properties screen is shown here. These values are obtained from a Properties Retrieval Job.

Example Eclipse preferences editing



Eclipse preferences can be created or configured using a job. This slide shows an example of an Eclipse Preferences Editing job.



Inventory jobs can be used to collect information about the set and state of installed OSGI bundles and services in the OSGi run-time.

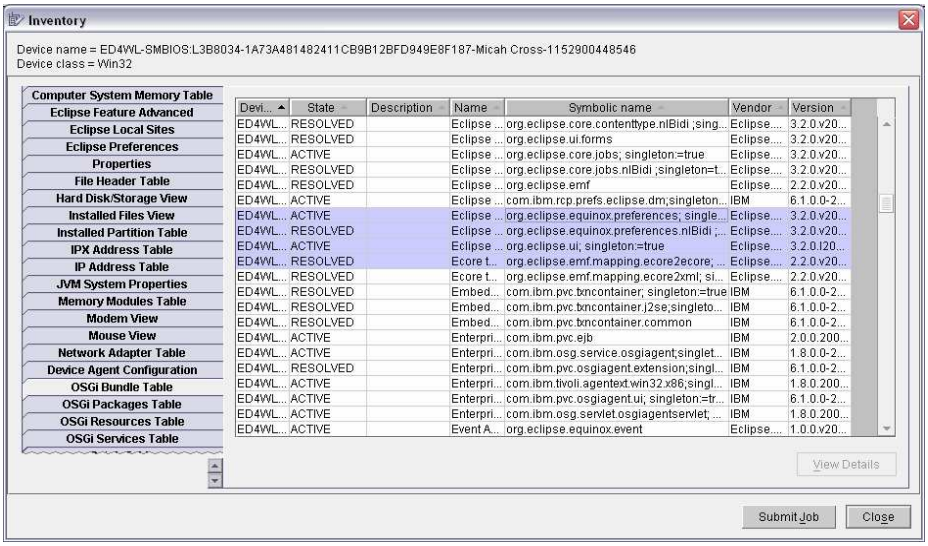
The bundle table shows information about the bundle including the bundle name, version, description, vendor and bundle state.

The package table provides the name and version of the packages.

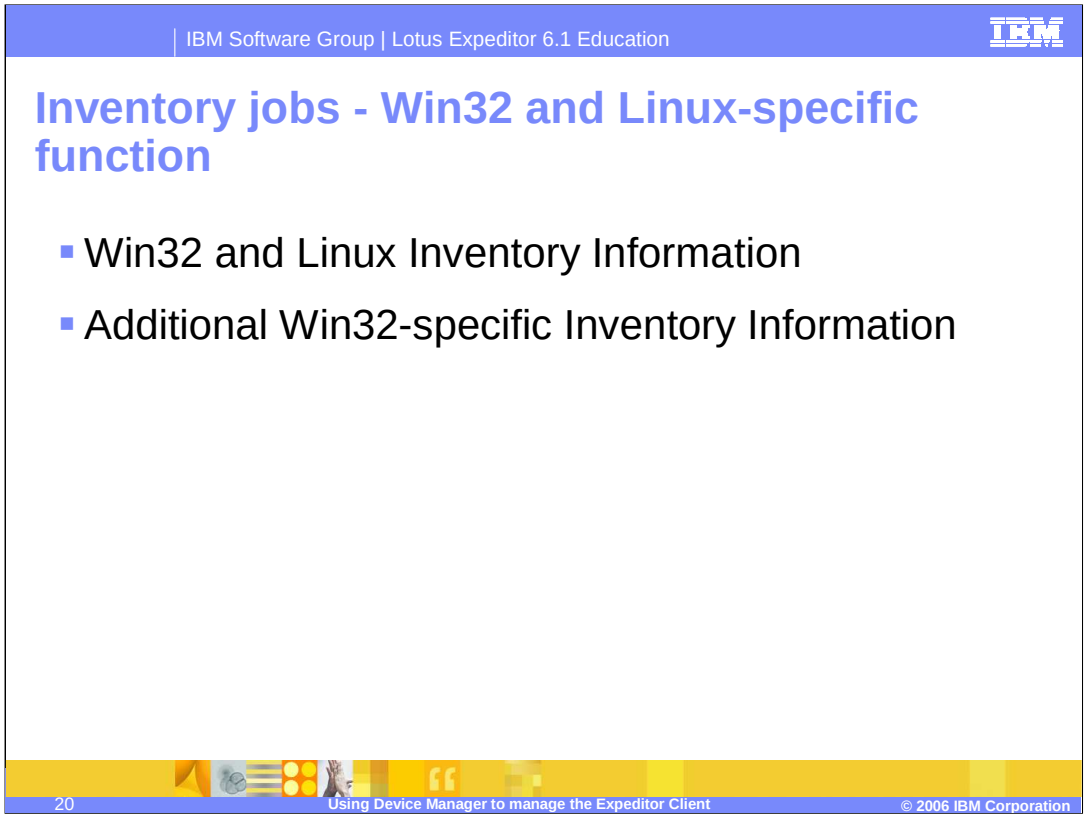
Service names are contained in the service table.

Finally, resource names and values can be viewed from the resource table.

Inventory jobs: Example OSGi bundle inventory



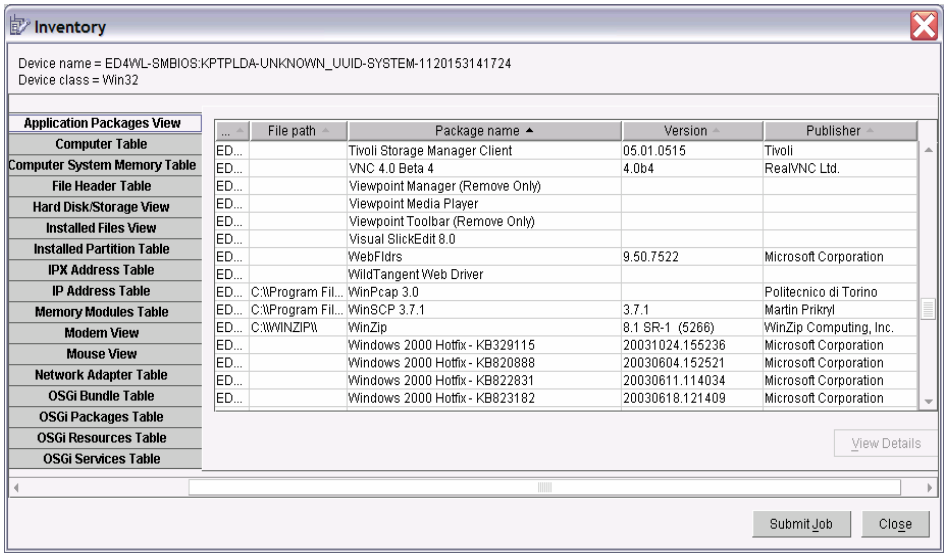
This slide shows an example of the OSGi Bundle Inventory view. Identifying information is displayed for each bundle.



Inventory jobs have some Windows 32 and Linux specific functions. They can be used to collect information about the set of native software, hardware and other information on the machine. This can include registered software, file scans, SMBIOS information, OS version, and hardware information.

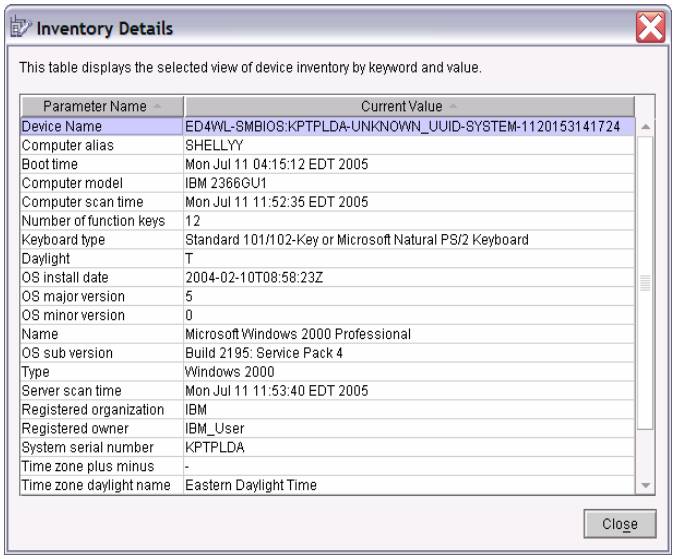
Also, on Windows 32 devices, information such as Windows services, user information, patch information, and registry entries is available.

Inventory jobs - Sample Win32 registered software



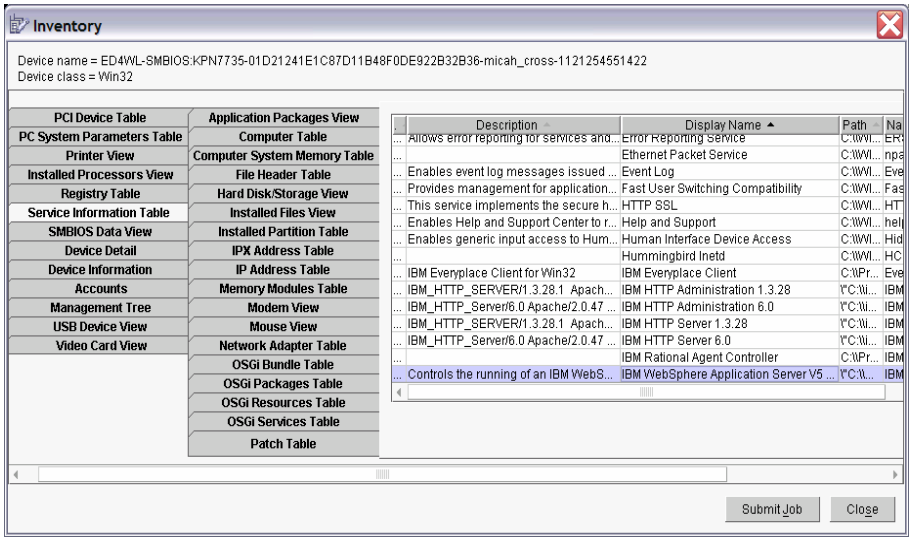
This view shows an example of a Windows 32 client inventory job. The registered software on the device, along with path, version and publisher information is shown in table form.

Inventory jobs - Sample computer OS/system inventory

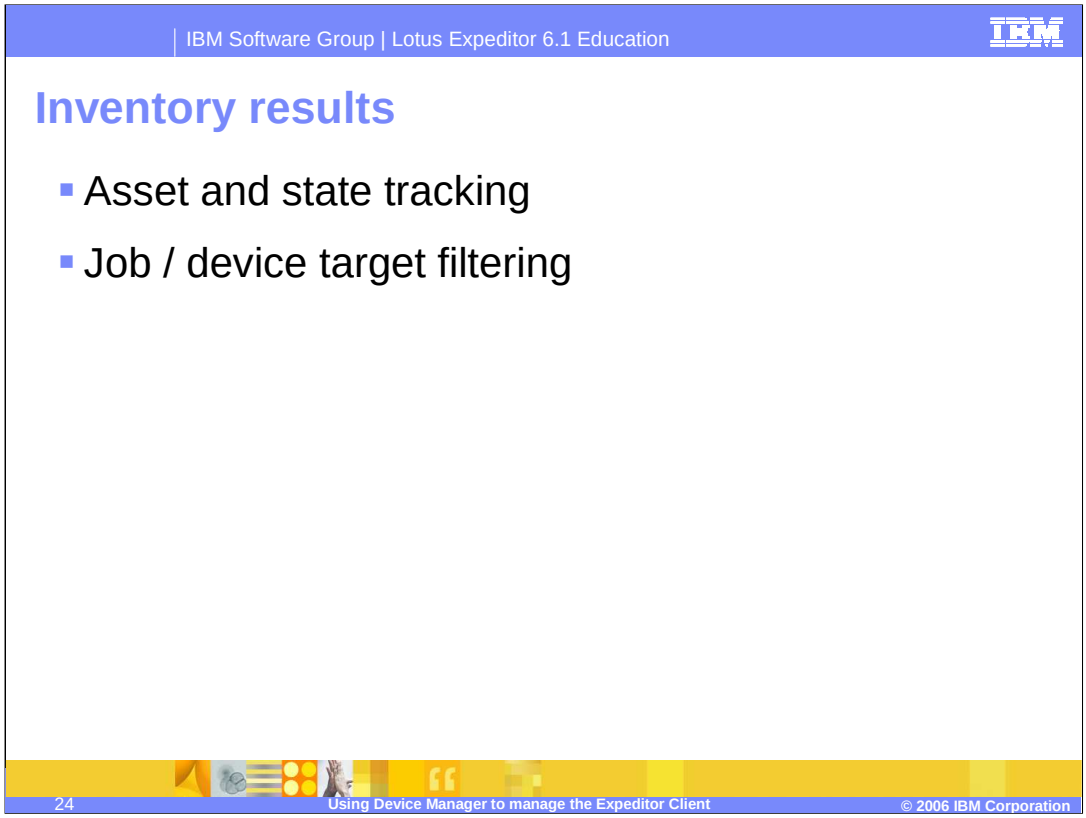


This slide shows the OS System Inventory detailed view of an Inventory job. Information about the operating system and machine is displayed.

Inventory jobs - Sample Win32 services inventory



Windows 32 services can be viewed from the Service Information Table on the Inventory screen.



Inventory results can be used for several potential applications, including asset or state tracking and job or device target filtering.

Automated customized scans can be scheduled to ease the burden of asset and state tracking. Inventory results can be used to create reports or automatic job submissions, depending on the scenario.

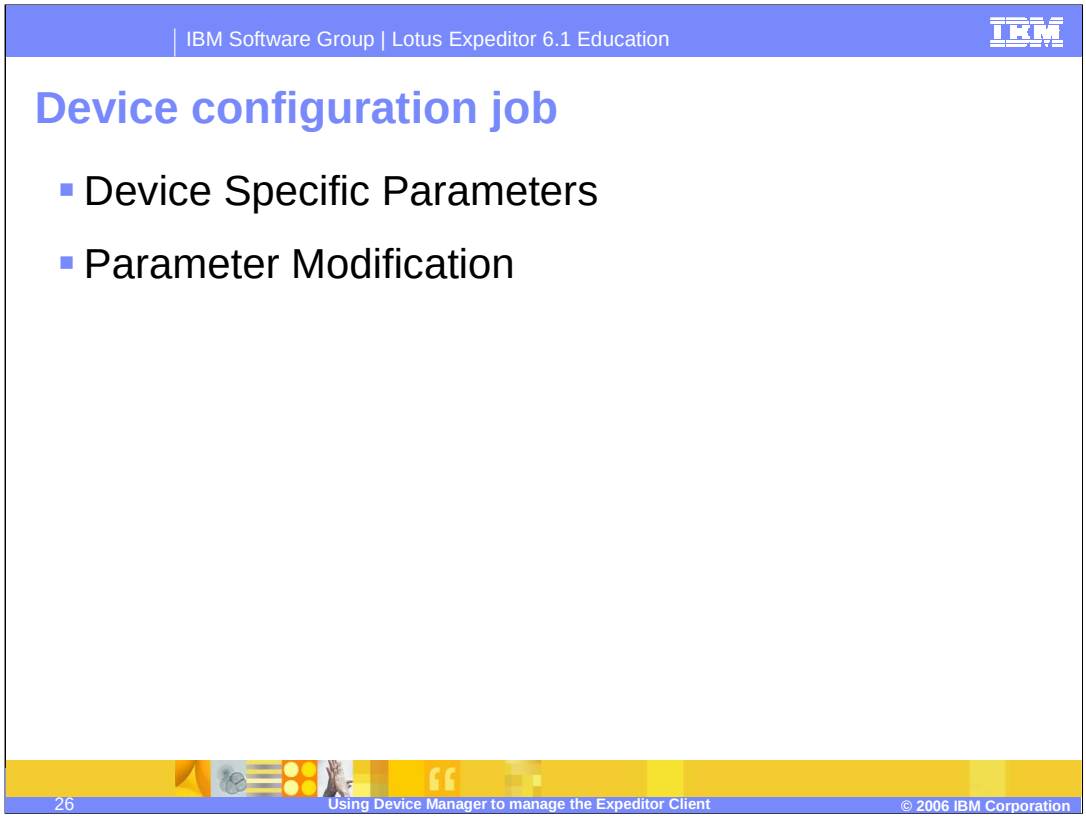
Any field data available through inventory can be used for mass device and job targeting.

An example would be patching a particular feature version on clients with an older version. Enrollment characteristics can be used to target unknown devices that match certain criteria.

A node discovery job recursively finds sub-nodes on the target devices based upon a target URI.

All the discovered sub-nodes are collected from the device and sent to the Device Manager server. The results can be stored in the Device Manager database. The search depth can be specified.

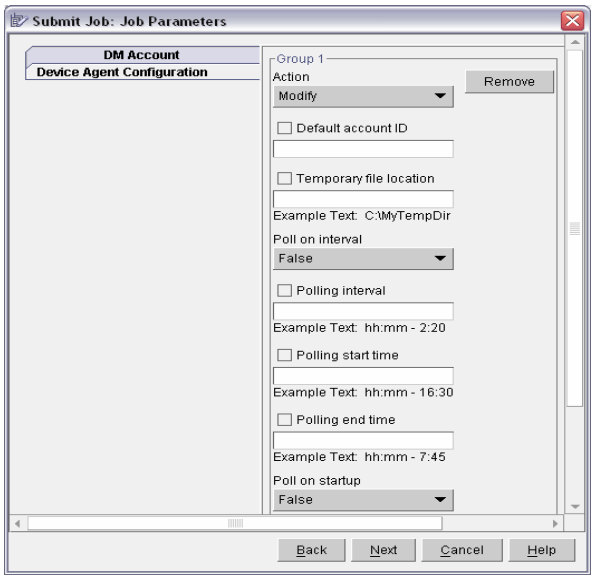
Jobs can be submitted that utilize the data collected by the node discovery job.



For every device, there are configuration parameters that are specific to that device. The configuration parameters identify the device with a device ID, manufacturer, model, and other parameters. The values for the configuration parameters are set for a device, a group of devices in the same device class, or all devices in the device class.

An administrator can modify the values of the configuration parameters using a Device Configuration job.

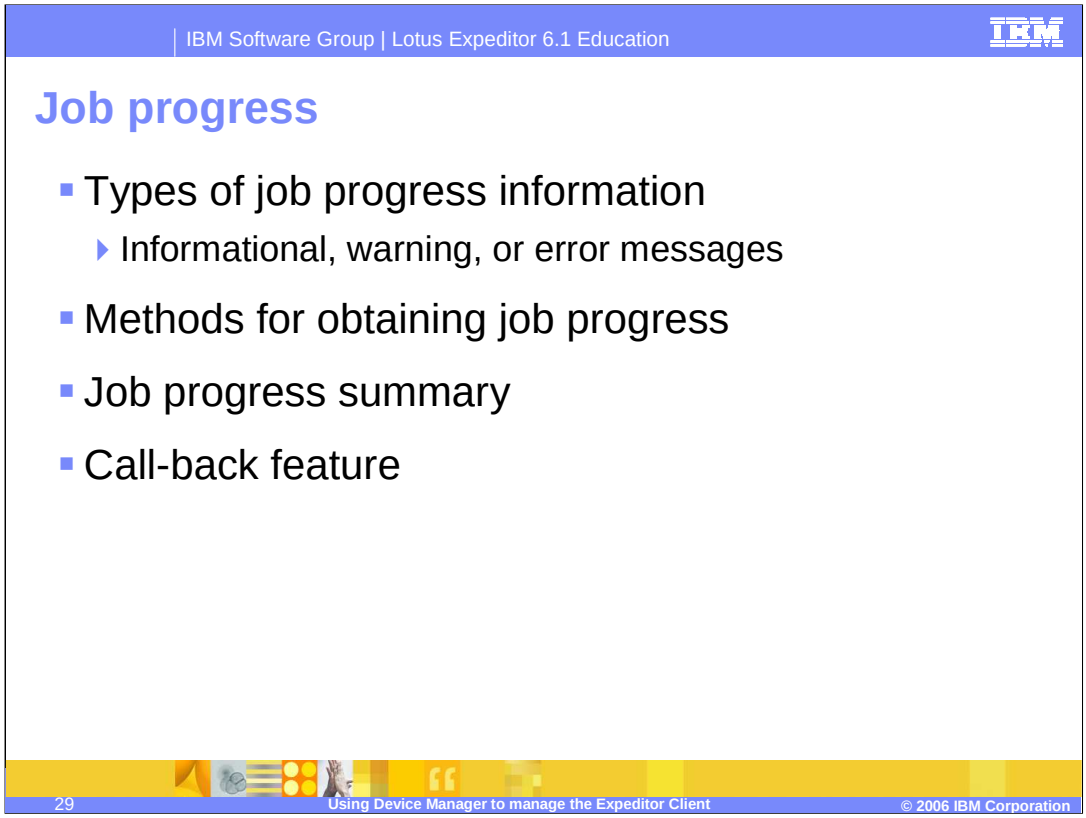
Device agent configuration



This slide shows an example of the device agent configuration screen. Device agent parameters can be modified using this view.

A run command job provides the ability to run a command on the device. The device agent sends status messages to standard output and standard error so appropriate administrative actions can occur. Jobs can be set to fail on non-zero return codes.

Examples of run command jobs include launching an installation and capturing command line output.



Information can be displayed about the progress of a job as it runs, or attempts to run, on its target devices.

This information includes any informational, warning, or error messages logged about the job processing on a target device.

This information can be obtained using the Device Manager console, the administration commands, and the Administration API.

A job progress summary for any job can also be displayed. The summary is most useful in understanding the progress of jobs submitted to multiple devices; that is, jobs submitted to a device class or to more than one device.

A call-back feature from the Device Manager server to the device agent is provided, so job events, such as when Device Manager starts processing a job, job completion status, and job expiration, can be monitored. The job events can be posted to a Java Message Service topic or with an HTTP GET command.

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:

Everyplace IBM Lotus WebSphere Workplace

Win32, Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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 or changes in the products or programs 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 (for example, 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.

This concludes the presentation.