Providing OSLC support
Providing OSLC support
Edition Notices

Before using this information and the product it supports, read the information in “Notices” on page 15.

This edition of *IBM Tivoli® System Automation Application Manager, Administrator's and User's Guide* applies to Version 4, Release 1, Modification 0 of IBM Tivoli System Automation Application Manager, program number 5724-S92, and to all subsequent releases and modifications of this product until otherwise indicated in new editions. IBM Tivoli System Automation Application Manager is the successor to the End-to-End Automation Management component of IBM Tivoli System Automation for Multiplatforms V2.3.

This edition replaces SC34-2587-03.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

IBM Deutschland Research and Development GmbH
Department 3248
Schoenaicher Str. 220
D-71032 Boeblingen
Federal Republic of Germany

FAX (Germany): 07031+16-3456
FAX (Other Countries): (+49)+7031-16-3456

Internet e-mail: eservdoc@de.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following information in your comment or note:

• Title and order number of this book
• Page number or topic related to your comment

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

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Figures ........................................ v
Tables .......................................... vii

Chapter 1. Integrating with JazzSM Administration Services ......... 1

Chapter 2. Integration scenario ......... 3
Registering resources ................. 3
Using REST web services ............. 4

Chapter 3. Configuring JazzSM Administration Services integration .... 5
Configuring the automation engine to register resources ................. 5
Installing the task bundle tsaControl ............... 5

Chapter 4. Administering registered JazzSM Administration Services resources ......... 7
Calculating the ResourceID from the ResourceKey ............... 7

Administering the tsaControl Task Bundle ............... 7
Operating the Administration Services user and command line interface ............... 8
Operating the Administration Services REST web service ............... 8
Example: Get the status information of an end-to-end resource using a PERL script ............... 9
Building a REST call to get all ResourceIDs ............... 10
Parsing the status information ............... 11
Parsing the resource details information ............... 12

Using IBM Support Assistant ............... 13
Installing IBM Support Assistant and the Tivoli System Automation for Multiplatforms plug-in ............... 13

Notices ............... 15

Trademarks ............... 16

Index ............... 17
Tivoli System Automation Application Manager v4.1 provides OSLC support: Providing OSLC support
Figures

1. JazzSM Administration Services integration scenario .............. 3
2. Registering resources for OSLC .............. 4
3. Using OSLC web services interfaces .............. 8
Tivoli System Automation Application Manager v4.1 provides OSLC support: Providing OSLC support
Tables

1. OSLC tab of the configuration utility . . . . 5
Chapter 1. Integrating with JazzSM Administration Services

System Automation Application Manager version 4.1 provides the capability to register resources through Administration Services of another JazzSM installation. This allows users to control resources from remote and get status information for System Automation Application Manager resources, which are defined in the end-to-end automation policy.

The Administration Services provides an interface to register and control resources and stores the data in the Registry Services. This data is compliant with the Open Services Lifecycle Collaboration (OSLC) standard. OSLC [http://open-services.net/] is an open standard by the OASIS consortium.

If the integration with JazzSM Administration Services is configured, the resources of the end-to-end automation policy are automatically registered. After registration these resources can be viewed and controlled by sending requests using a REST interface.

System Automation Application Manager supports the following actions for registered resources:
- List resources from the end-to-end automation policy.
- Get the status of a resource from the end-to-end automation policy.
- Send a start request to a resource of the end-to-end automation policy.
- Send a stop request to a resource of the end-to-end automation policy.

Benefits of the OSLC integration:
- Any program or script can be extended with code that is building the https connection and sending the REST calls.
- By calling the REST interface it is not necessary for a user to have the authority to log on to the operating system running the System Automation Application Manager.
- A REST call does not need to start a JVM, multiple single actions can send requests faster.
Chapter 2. Integration scenario

If you want to integrate with JazzSM Administration Services, a setup of two WebSphere installations is required:

1. WebSphere Application Servers hosting System Automation Application Manager components, running with a 32 Bit JDK.
2. WebSphere Application Server hosting the Administration Services and Registry Services components, running with a 64 Bit JDK.

Both servers needs access to a DB2 database, which may be on any system. This database can be the same for both components.

![Diagram of JazzSM Administration Services integration scenario](image)

**Figure 1. JazzSM Administration Services integration scenario**

**Note:** Registry Services and System Automation Application Manager do not need to share a DB2 database.

For installing the Administration and Registry Services, refer to [http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/install/psc_c_install.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/install/psc_c_install.html).

### Registering resources

Resource can only be created or deleted if an end-to-end automation policy is activated or deactivated. Then the new state of the end-to-end resource topology is registered within the Registry Services by calling the Administration Services. Following this pattern ensures, that the resource registry and the end-to-end automation policy resources stay synchronized.

To activate resource registration, provide the necessary configuration parameters in the OSLC tab of the configuration utility, see “Registering resources.”

If you want to administer registered resources, an Administration Services task bundle has to be deployed. These task bundles are plug-ins that define actions for resources of a specific type and contain the business logic of the action. Task bundles have to be installed manually (see “Installing the task bundle tsaControl” on page 5) and are associated to registered resources if their target type matches the resource type of a resource.
1. A new end-to-end policy gets activated, for example end-to-end resources change.
2. The automation engine automatically registers the resources at the Administration Services using REST.
3. For each resource a task bundle is deployed to allow status retrieval and to send start and stop requests.

Using REST web services

After the registration, end-to-end automation resources can be instrumented using either one of the following interfaces provided by Administration Services:

- **Administration page in DASH**: [http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/admin/as_c_ui_oview.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/admin/as_c_ui_oview.html)
- **Command line interface**: [http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/config/as_c_config_client_cli.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/config/as_c_config_client_cli.html)
- **REST web service**

The following actions are available to administer end-to-end automation resources:

- Get status
- Send start request
- Send stop request
- List end-to-end resources
Chapter 3. Configuring JazzSM Administration Services integration

Configuring the automation engine to register resources

The configuration of the automation engine is necessary to activate the JazzSM Administration Services integration. You can also define the parameter of the connection to the other WebSphere Application Server, for example the location, port, or which authority is required to register.

Configure the automation engine for resource registration, by invoking the configuration utility with the `cfgeezdmn` command. The tab **OSLC** contains the relevant configuration parameters. For silent configuration the parameters are configured as part of the common configuration settings. The parameters names, their type and function are listed in the following table:

Table 1. OSLC tab of the configuration utility

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslc-enable</td>
<td>boolean</td>
<td>Decides whether resources will be registered at Administration Services.</td>
</tr>
<tr>
<td>oslc-server-name</td>
<td>String, hostname or IP</td>
<td>The hostname or IP of the system hosting the Administration Services.</td>
</tr>
<tr>
<td>oslc-server-port</td>
<td>Integer, port</td>
<td>The WC_DEFAULTHOST_SECURE port of WebSphere Application Server hosting the Administration Services.</td>
</tr>
<tr>
<td>oslc-registry-userid</td>
<td>String userID</td>
<td>The WebSphere Application Server userID which executes the registry call. The RegistryAdminRole is required.</td>
</tr>
<tr>
<td>oslc-registry-password</td>
<td>String, password</td>
<td>The password of the user defined by the oslc-registry-userid.</td>
</tr>
<tr>
<td>oslc-ssl-truststore</td>
<td>String, filename</td>
<td>Filename of a truststore file of type JKS which contains the certificate of the WebSphere Application Server hosting the Administration Services.</td>
</tr>
<tr>
<td>oslc-ssl-truststore-password</td>
<td>String, password</td>
<td>Password to access the file defined by oslc-ssl-truststore.</td>
</tr>
</tbody>
</table>

The truststore can be created by importing the certificate and defining a new filename by using the Java Keytool with the following command:

```
keytool -import -alias oslc -file $certificateFile -keystore oslcKeystore.jks
```

After configuration restart or refresh the automation engine.

Installing the task bundle tsaControl

Upon registration, resources in the Registry Services do not have any associated actions. Installing the appropriate task bundle associates the resources of a specific type to the actions defined the task bundle.
Since the application logic of the task bundles needs to establish an https-connection to the WebSphere Application Server hosting System Automation Application Manager, it is required to provide the certificate to the local openssl installation.

System Automation Application Manager provides the task bundle tsaControl as a ZIP archive, located in the subfolder integration on the installation medium.

To install tsaControl, execute the following steps:
1. Install openssl, if not already installed.
2. Copy the ZIP archive to the machine where Administration Services is installed.
3. Install the task bundle using the Administration Services CLI using the following command:
   
   ```
   $jazzSMInstallDir/admin/bin/adminservices.sh -installTasks $pathOfTaskBundle/tsaControl.zip
   ```

4. Add the certificate of the WebSphere Application Server hosting System Automation Application Manager to the OPENSSLDIR of the openssl installation. This information can be retrieved by executing the command:
   
   ```
   openssl version -a
   ```
Chapter 4. Administering registered JazzSM Administration Services resources

Calculating the ResourceID from the ResourceKey

The Administration Services Resource Registry needs an unique identifier to distinguish resources. The Administration Services do not allow characters like blanks or slashes in their ID. For System Automation Application Manager, the ResourceKey serves as the unique identifier. As characters like blanks or slashes are allowed in System Automation Application Manager, the ResourceKey cannot be used as identifier in the Administration Services Resource Registry.

Perform the following steps to transfer the ResourceKey into a ResourceID for the Administration Services Resource Registry:

1. Calculate the SHA-1 checksum of the ResourceKey (String literal, UTF-8 encoded).
2. Execute a base16 (hexadecimal) encoding of the SHA-1 checksum.
3. If the base16 encoder produced lower case letters, convert to uppercase letters.

The result is a String containing exactly 40 characters of the character class [0-9A-F].

Administering the tsaControl Task Bundle

The tsaControl task bundle associates itself automatically to any resource of the type 'TSAResource' registered via Admin Services. Since the Automation Engine registers all resources from the End-to-End Automation Policy with this resource type, the association will always be established.

When called the action requires the following parameters:

hostname
  Hostname or IP of the WebSphere Application Server hosting System Automation Application Manager.

port
  Default host secure port of the WebSphere Application Server hosting System Automation Application Manager.

user
  EEZ user that has the authority to perform the actions.

password
  Password of the EEZ user.

action
  One action from the list:
  • status
  • start
  • stop

resourceKey
  EEZResourceKey in this format:
  $domainName:$resourceClass:$resourceName[:$ flaDomain]
For example, FriendlyE2E:ResourceReference:Ref2:cluster1 or FriendlyE2E:ResourceGroup:RG1

**comment**

A comment (optional). If the performed action is 'status', the comment will be discarded.

---

### Operating the Administration Services user and command line interface

If you want to find out how to operate the Administration Services user interface, refer to [http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/admin/as_c_ui_oview.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/admin/as_c_ui_oview.html).

If you want to find out how to operate the Administration Services command line interface, refer to [http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/config/as_c_config_client_cli.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0.2/config/as_c_config_client_cli.html).

---

### Operating the Administration Services REST web service

A client sends a REST call to the Administration Services using his REST interfaces. If the call is a request, the Administration Services will send a reply to indicate whether the request has been accepted. If the call is a request to invoke the tsaControl task bundle, the application logic of the task bundle contacts the DASH of the System Automation Application Manager and executes the desired action.

A second call by the client is made to retrieve the result. The answer from Administration Services will contain the data returned by the tsaControl task bundle.

![Diagram showing the process of using OSLC web services interfaces](image)

*Figure 3. Using OSLC web services interfaces*

1. An OSLC compatible client polls for the available resources of type TSAResource.
2. Administration Services returns a list of resources from the registry.
3. OSLC client determines the available actions for the resource.
4. OSLC issues an action **status**, **start**, or **stop** against the TSAResource using the task bundle.
5. The task bundle issues an appropriate call to the data provider registered in DASH.

If you want to use the REST interface, execute the following steps:

1. Select the tool to send the REST call. This may be a Java® program, a PERL script or a browser plug-in. Many programming or scripting languages provide methods and modules to generate http-requests of any kind.

2. Create an HTTP request with the appropriate method GET or POST. Listing resources and retrieving results requires GET, invoking an action requires POST.

3. Attach basic authorization to the HTTP request, using OSLC username and OSLC password. In order to successfully execute the call the user must have proper authorization in the JazzSM Administration Services context.

4. Configure the client to accept the certificate of the WebSphere Administration Server running the Administration Services. Methods include using a KeyStore or leveraging the openssl configuration.

5. The documentation of the JazzSM Administration Services REST interfaces shows the URLs, methods, expected parameters, and necessary authentication. For more information, refer to Administration Services Provider REST Interfaces.

6. If the HTTP request is a POST request the content needs to be defined. For task bundle invocation the parameters listed in “Administering the tsaControl Task Bundle” on page 7 need to be added to the RDF/XML document to be sent. Each parameter requires a section like the following:

   <rdf:Description rdf:nodeID="A7">
   <rdf:value>$variableValue</rdf:value>
   <$oslc:name>variableName</oslc:name>
   <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>

7. Build the REST call and send it. If the call was a request, the immediate answer will confirm the acceptance of the request only. A second call to retrieve the result must be made to access the results of the executed action.

Example: Get the status information of an end-to-end resource using a PERL script

1. Get the following input:
   - System Automation Application Manager: hostname, port, user, password
   - OSLC: hostname, port, user, password, ResourceKey

2. Calculate the ResourceID from the ResourceKey. For more information, refer to “Calculating the ResourceID from the ResourceKey” on page 7.

3. Build the HTTP Request with the following parameters:
   a. URL:

   ```perl
   my $url = https://$oslcHost:$oslcPort/admin/services/tasks/request/tsaControl/$resourceID
   ```

   b. Create a HTTP POST Request object:

   ```perl
   my $request = HTTP::Request->new(POST => $url);
   ```

   c. Define application/rdf+xml as the content type:

   ```perl
   $request->header('Content-Type' => 'application/rdf+xml');
   ```

   d. Authorization with $oslcUser and $oslcPassword:

   ```perl
   $request->authorization_basic($oslcUser, $oslcPass);
   ```

   e. Create the user agent:

   ```perl
   my $ua = LWP::UserAgent->new( (... ) );
   ```
f. Define content:

```
my $postData = "<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:oslc="http://open-services.net/ns/core#"
   xmlns:j.0="http://open-services.net/ns/auto#" >
   <rdf:Description rdf:nodeID="A0">
     <rdf:value>$saamHost</rdf:value>
     <oslc:name>hostname</oslc:name>
     <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A1">
     <rdf:value>$saamPort</rdf:value>
     <oslc:name>port</oslc:name>
     <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A2">
     <rdf:value>$saamUser</rdf:value>
     <oslc:name>user</oslc:name>
     <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A3">
     <rdf:value>$saamPass</rdf:value>
     <oslc:name>password</oslc:name>
     <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A5">
     <rdf:value>status</rdf:value>
     <oslc:name>action</oslc:name>
     <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A7">
     <rdf:value>$resourceKey</rdf:value>
     <oslc:name>resourceKey</oslc:name>
     <rdf:type rdf:resource="http://open-services.net/ns/auto#ParameterInstance"/>
   </rdf:Description>
   <j.0:executesAutomationPlan rdf:resource="https://$oslcHost:$oslcPort/admin/services/tasks/actions/tsaControl/automation"/>
   <j.0:inputParameter rdf:nodeID="A0"/>
   <j.0:inputParameter rdf:nodeID="A1"/>
   <j.0:inputParameter rdf:nodeID="A2"/>
   <j.0:inputParameter rdf:nodeID="A3"/>
   <j.0:inputParameter rdf:nodeID="A4"/>
   <j.0:inputParameter rdf:nodeID="A5"/>
   <j.0:inputParameter rdf:nodeID="A6"/>
   <j.0:inputParameter rdf:nodeID="A7"/>
   </rdf:RDF>";
```

g. Attach the content to the request:
```
$request->content($postData);
```
h. Send the request
```
my $rsp = $ua->request($request);
```
i. After the request has been processed by Administration Services the result can be retrieved by sending a GET request to the URL:
```
https://$oslcHost:$oslcPort/admin/services/tasks/result/tsaControl/$resourceID
```

**Building a REST call to get all ResourceIDs**

Use the following code:
URL url = new URL("https://" + hostname + ":" + port + "/admin/services/adminresources/collection");
HttpsURLConnection.setDefaultHostnameVerifier(hv);
HttpsURLConnection urlConn = (HttpsURLConnection) url.openConnection();
FileInputStream fisKeyStore = new FileInputStream(keyStoreLocation);
keyStore.load(fisKeyStore, keyStorePassword);
// Setup a SSL socket factory via SSLContext and TrustManagerFactory
String userCredentials = user + ":" + password;
String basicAuth = "Basic " + new String(DatatypeConverter.printBase64Binary(userCredentials.getBytes()));
urlConn.setRequestProperty("Authorization", basicAuth);
urlConn.setRequestMethod("GET");
urlConn.setAllowUserInteraction(false); // no user interaction
urlConn.setRequestProperty("Content-type", "application/rdf+xml");
urlConn.setRequestProperty("Content-length", "/admin/services/adminresources/collection".getLength());
urlConn.setRequestProperty("Content-Language", "en-US");
urlConn.setUseCaches(false);
urlConn.setDoOutput(true); // want to send
urlConn.setDoInput(true); // want to receive

// Send request
DataOutputStream wr = new DataOutputStream(urlConn.getOutputStream());
wr.writeBytes(rdf);
wr.flush();
wr.close();

--- Parsing the status information ---

You can retrieve the status information using the GET command in the RDF of the result. The output appears twice, in a j.o:messageDetails section and in rdf:Description/rdf:value and will be formatted depending on the performed action and whether the action was a success or a failure:

Output, if an error occurred:
--- SA AM Status Error BEGIN ---
ResourceKey: $resourceKey
Error: $message
HTTP Response: $httpResponseCode $httpResponseMessage
--- SA AM Status Error END ---

HTTP response codes and messages are defined in RFC 2616 of the IETF:
https://www.ietf.org/rfc/rfc2616.txt

Output for status information:
--- SA AM Status Information BEGIN ---
ResourceKey: $resourceKey
DesiredState: $desiredState
ObservedState: $observedState
OperationalState: $displayStringOfOperationalState
CompoundState: $compoundState
--- SA AM Status Information END ---

If the action was start or stop and no failure occurred, then no additional information is printed.

Any parsing should look for the header line and then parse the following lines until the footer line has been reached. The ResourceKey is shown again for easier parsing and matching or building data structures. The header and footer lines can work as eye catcher for a parser to start and stop its work. All output before the header or after the footer can possibly be discarded and all important information is between those two.
Examples:
1. In PERL use the regular expression `/^([^:])\s+(.+)$/` and save \$1 as key and \$2 as value into a hashmap. This method works even if the order changes.
2. Use awk '{ print $2; }' to print the status information.

Parsing the resource details information

Viewing one resource with details via AdminServices REST API will return the following RDF. Highlighted is the section with details about the resource:

Sample Answer

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:oslc="http://open-services.net/ns/core#"
  xmlns:art="http://jazz.net/ns/ism/admin#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:art="http://jazz.net/ns/ism/admin#AdministrableResource"
  xmlns:crtv="http://open-services.net/ns/crtv#"
  xmlns:oslc="http://open-services.net/ns/core#">
  <art:AdministrableResource
    rdf:about="https://saxb56e.boeblingen.de.ibm.com:16311/admin/services/adminresources/resource/362E96078B8A8F33E462D4848097BB01ED088B3">
    <oslc:domain rdf:resource="http://jazz.net/ns/ism/admin/1.0/"/>
    <rdf:type rdf:resource="http://jazz.net/ns/ism/admin#AdministrableResource"/>
    <dcterms:identifier>362E96078B8A8F33E462D4848097BB01ED088B3</dcterms:identifier>
    <dcterms:description>This is the reference with the name Enterprise DB2</dcterms:description>
    <crtv:version>1.0</crtv:version>
    <crtv:parentServiceInstance rdf:resource="http://open-services.net/ns/crtv#NULL"/>
    <dcterms:subject>AdminServices</dcterms:subject>
    <art:healthStatus
      rdf:resource="https://saxb56e.boeblingen.de.ibm.com:16311/admin/services/adminresources/status/362E96078B8A8F33E462D4848097BB01ED088B3"/>
    <art:relatedJobs
    <art:AdministrableResource/>
  </art:AdministrableResource>
</rdf:RDF>
```
Using IBM Support Assistant

IBM Support Assistant is a free, stand-alone application that you can install on any workstation. IBM Support Assistant saves you time searching product, support, and educational resources and helps you gather support information when you need to open a problem management record (PMR) or Electronic Tracking Record (ETR), which you can then use to track the problem.

You can then enhance the application by installing product-specific plug-in modules for the IBM products you use. The product-specific plug-in for Tivoli System Automation for Multiplatforms provides you with the following resources:
- Support links
- Education links
- Ability to submit problem management reports
- Capability to collect traces

Installing IBM Support Assistant and the Tivoli System Automation for Multiplatforms plug-in

To install the IBM Support Assistant V4.1, complete these steps:
- Go to the IBM Support Assistant Web Site: www.ibm.com/software/support/isa/
- Download the installation package for your platform. Note that you will need to sign in with an IBM user ID and password (for example, a MySupport or developerWorks® user ID). If you do not already have an IBM user ID, you may complete the free registration process to obtain one.
- Uncompress the installation package to a temporary directory.
- Follow the instructions in the Installation and Troubleshooting Guide, included in the installation package, to install the IBM Support Assistant.

To install the plug-in for Tivoli System Automation for Multiplatforms, complete these steps:
1. Start the IBM Support Assistant application. IBM Support Assistant is a Web application that is displayed in the default, system configured Web-browser.
2. Click the Updater tab within IBM Support Assistant.
3. Click the New Products and Tools tab. The plug-in modules are listed by product family.
5. Select the features you want to install and click Install. Be sure to read the license information and the usage instructions.
6. Restart IBM Support Assistant.
Tivoli System Automation Application Manager v4.1 provides OSLC support: Providing OSLC support
Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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.

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
Mail Station P300
2455 South Road
Poughkeepsie New York 12601-5400
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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamaoto-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL
Index

N
notices  15

O
OSLC
  configuring
    automation engine  5
    configuring JazzSM  5
    installing tsaControl  5
    registering resources  3
    ResourceID  7
    ResourceKey  7
    REST web services  4

T
trademarks  16
Tivoli System Automation Application Manager v4.1 provides OSLC support: Providing OSLC support