SWG - TIVOLI

IBM Tivoli Directory Integrator
version 6.0
WS Provisioning in ITDI

Issue Date: 2004.08.27

Revision Status: Draft

Document Revision History

Version	Date	Created by	Short Description
1.0	2004.08.27	Stanislav Ovcharov	Initial Version

WS-Provisioning in IBM Tivoli Directory Integrator

Overview

The WS-Provisioning support is based on the IBM Tivoli Directory Integrator Web Service components. The WS-Provisioning package "wsprov.jar" JAR file contains the compiled Java classes which implement the data types used by WS-Provisioning. These Java classes and the Web Services components will help you to develop or connect to a WS-Provisioning implementation.

There are three Java packages in "wsprov.jar":

- api._1._0.provisioning.ws.names.ibm – implements types defined in the "api.xsd" schema, which is part of the WS-Provisioning specification
- core._1._0.provisioning.ws.names.ibm – implements types defined in the "core.xsd" schema, which is part of the WS-Provisioning specification
- notify._1._0.provisioning.ws.names.ibm – implements types defined in the "notification.xsd" schema, which is part of the WS-Provisioning specification

WS-Provisioning DRAFT Version 0.7 (01 October 2003) is supported by this package.

Mapping of Java class names to WS-Provisioning XSD types

The name of each Java class matches the name of a WS-Provisioning schema type.
For example, the "api._1._0.provisioning.ws.names.ibm.ListTargetsRequestType" Java class implements the type "ListTargetsRequestType" defined in the "urn:ibm:names:ws:provisioning:0.1:api" namespace.

The "wsprov.jar" file contents

The "wsprov.jar" file contains Java classes generated using Axis 1.1 WSDL2Java tool. There were errors in some of the Java files generated by WSDL – these files were edited manually to fix these errors and then all sources were compiled and packed into "wsprov.jar".

Deployment

You need to copy the "wsprov.jar" file into the "jars" folder under the root folder where the IBM Tivoli Directory Integrator is installed. Next time the IBM Tivoli Directory Integrator Server is started it will load the WS-Provisioning classes and they can be used in your AssemblyLines.

Note: The Java sources of the "wsprov.jar" file are included in the "src" folder of this package. You can use them for reference and also you could modify and customize them on your own.

Examples

- Create the input for the "listTargets" operation in Javascript:

```
var listTargetsInput = new
Packages.api._1._0.provisioning.ws.names.ibm.ListTargetsRequestType();
```

When serialized this "`listTargetsInput`" variable will look like this:

```
<ListTargetsRequest xsi:type="ns1:ListTargetsRequestType"
xmlns="urn:ibm:names:ws:provisioning:0.1:api"
xmlns:ns1="urn:ibm:names:ws:provisioning:0.1:api"/>
```

- Create the output for the "listTargets" operation in Javascript:

```
var set = new
Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningTargetSetType();
var provTargetType = new
Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningTargetType();
var provTargets =
java.lang.reflect.Array.newInstance(provTargetType.getClass(), 1);
var id = new
Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningIdentifierType();
id.setName("WindowsNTServerAccount");
provTargets[0] = provTargetType;
provTargets[0].setIdentifier(id);
var provTargetSchema = new
Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningTargetSchema();
var schema = java.lang.reflect.Array.newInstance(provTargetSchema.getClass(),
1);
schema[0] = provTargetSchema;
schema[0].setLocation("http://myhost.com/myschema.xsd");
provTargets[0].setSchema(schema);
set.setProvisioningTarget(provTargets);
var response = new
Packages.api._1._0.provisioning.ws.names.ibm.ListTargetsResponseType();
response.setTargets(set);

var result = new
Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningIteratedResultType(
);
result.setSize(1);
var status = new
Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningRequestStatusType()
;
status.setCode(Packages.core._1._0.provisioning.ws.names.ibm.ProvisioningStat
usCode.success);
result.setStatus(status);
response.setResult(result);
```

When serialized this "`response`" variable will look like this:

```
<ns2:arg0 xsi:type="ns1:ListTargetsResponseType"
xmlns:ns1="urn:ibm:names:ws:provisioning:0.1:api"
xmlns:ns2="urn:ibm:names:ws:provisioning:0.1:psp">
   <ns1:targets xsi:type="ns3:ProvisioningTargetSetType"
xmlns:ns3="urn:ibm:names:ws:provisioning:0.1:core">
    <ns3:ProvisioningTarget xsi:type="ns3:ProvisioningTargetType">
     <ns3:identifier xsi:type="ns3:ProvisioningIdentifierType"
name="WindowsNTServerAccount"/>
     <ns3:schema xsi:type="ns3:ProvisioningTargetSchema"
location="http://myhost.com/myschema.xsd"/>
    </ns3:ProvisioningTarget>
   </ns1:targets>
   <ns1:result xsi:type="ns4:ProvisioningIteratedResultType"
remaining="0" size="1"
xmlns:ns4="urn:ibm:names:ws:provisioning:0.1:core">
    <ns4:status xsi:type="ns4:ProvisioningRequestStatusType">
     <ns4:code xsi:type="ns4:ProvisioningStatusCode">
      <ns4:value xsi:type="xsd:string">success</ns4:value>
     </ns4:code>
    </ns4:status>
   </ns1:result>
  </ns2:arg0>
```