OPSWARE INC
Automating IT™

Opsware System Web Services API 2.0 Guide

Corporate Headquarters

599 North Mathilda Avenue Sunnyvale, California 94085 U.S.A.
T + 1 408.744.7300 F +1 408.744.7383 www.opsware.com

Table Of Contents

Preface

Document Date: 1/10/05

Welcome to the Opsware System Web Services API 2.0I – a standards-based programming environment built on open industry standards such as SOAP (Simple Object Access Protocol) and WSDL (Web Services Definition Language).

The Opsware System Web Services API 2.0 enables the integration of applications and other systems with the [type product name here] versions 4.6 and 4.7. This broadens the scope of how IT can use the [type product name here] to achieve operational goals.

About this Guide

This guide describes how to use the Opsware System Web Services API 2.0, starting with an overview of the API. It provides information about using the API to manage servers, reconcile servers (install or remove software), manipulate the software node tree, and execute automated scripts.

Audience for this Guide

This guide is intended for software developers who will be writing programs using the Opsware System Web Services API 2.0.

To use the Opsware System Web Services API 2.0, you should:

- Be familiar with Opsware System (have taken the Opsware Fundamentals course or have equivalent experience).
- Have experience in programming a client application that interacts with a Web Service in one of the languages (Java, Python, Perl, C#) that the Opsware System Web Services API 2.0 supports.

Contents of this Guide

This guide contains the following chapters:

- **Chapter 1:** Overview - provides an overview description of the Opsware System Web Services API 2.0, including architecture diagrams and descriptions of the sample client programs
- **Chapter 2:** Set Up - discusses software requirements, security, exception handling, logging

Chapters 3 - 12 have reference information for each SOAP operation in a given Web Service. Each chapter begins with a brief overview of the Web Service, followed by the sections that describe the results and arguments of every operation. (See "How the Chapters on the Web Service Operations Document the WSDL Files" on page xiv.)

- **Chapter 3:** Command Session Web Service Operations Reference
- **Chapter 4:** Custom Attribute Web Service Operations Reference
- **Chapter 5:** Customer Web Service Operations Reference
- **Chapter 6:** Distributed Script Web Service Reference
- **Chapter 7:** Facility Web Service Operations Reference
- **Chapter 8:** Node Web Service Operation Reference
- **Chapter 9:** Platform Web Service Operations Reference
- **Chapter 10:** Security Web Service Operations Reference
- **Chapter 11:** Server Web Service Operations Reference
- **Chapter 12:** Software Web Service Operations Reference
- **Chapter 13:** Complex Data Types Reference - provides reference information for the SOAP complex types used by the Opsware System Web Services API 2.0. (See "How the Chapter on the Complex Data Types Reference Documents the WSDL Files" on page xv.)

Conventions in this Guide

This guide uses the following typographical and formatting conventions.

NOTATION	DESCRIPTION
Bold	Defines terms.
Italics	Identifies guide titles and provides emphasis.

NOTATION	DESCRIPTION
`Courier`	Identifies text of displayed messages and other output from Opsware programs or tools.
Courier Bold	Identifies user-entered text (commands or information).
Courier Italics	Identifies variable user-entered text on the command line or within example files.

Icons in this Guide

This guide uses the following iconographic conventions.

ICON	DESCRIPTION
📄	This is a note. This icon identifies especially important concepts that warrant added emphasis.
☑	This is a requirement. This icon identifies a task that must be performed before an action under discussion can be performed.
💡	This is a tip. This icon identifies information that can help simplify or clarify tasks.
⚠	This is a warning. This icon is used to identify significant information that must be read before proceeding.

Documentation Supporting the Opsware System Web Services API 2.0

The documentation set supporting the API includes:

- This reference guide – *Opsware System Web Services API 2.0 Guide*
- The developer syntax docs for each client language – Java, Python, Perl, C#

For more information, see "About the Developer Syntax Docs" on page 10.

- The Web Service Definition Language (WSDL) files – one WSDL file per Web Service (ten WSDLs total)

  The WSDL file is the single most accurate standard-compliant description of the Web Services. It includes definitions of the data structures in the form of XML schemas, the objects that can be sent and retrieved, the operations exposed in the interface, and their corresponding input and output parameters. The section "How to Read this Guide" on page xiv describes how this guide documents the operations defined in the WSDL files.

- The source code for a sample client application for each supported language

  The sample client applications show how to connect to the Web Service and how to invoke most of the getter operations, including those that use filter arguments for retrieving information.

- The README for the sample application for each client platform that describes how to set up and run the sample application for that platform

In addition to the supporting documentation set, Opsware Inc. provides the following resources for developers using the Opsware System Web Services API 2.0:

- Client stubs (See "Client Architecture" on page 7.)
- Proxy modules (See "Supported Client Languages and Technologies" on page 9.)
- A static WSDL file for each Web Service

How to Read this Guide

The organization of this guide mirrors the underlying structure of the Opsware System Web Services API 2.0. The reference chapters of this guide describe every Web Service operation and most of the complex SOAP data types used by the operations.

How the Chapters on the Web Service Operations Document the WSDL Files

In this guide, reference chapters document the operations for each Web Service and corresponding WSDL file. To find out the syntax of an operation in a particular language (Java, Python, Perl, C#), refer to the developer syntax docs (such as the javadocs).

For example, The Server Web Service Operations Reference chapter documents the `getIDs` operation in a language-neutral manner by describing the operation result and argument as SOAP data types. The `ServerWebService.wsdl` file specifies the `getIDs` operation as follows:

```
<operation name="getIDs"  >
  <input message="tns:getIDs" />
  <output message="tns:getIDsResponse" />
</operation>
. . .
<operation name="getIDs" >
<soap:operation
soapAction=""
style="rpc" />
<input>
  <soap:body use="encoded"
  namespace="http://www.opsware.com/osapi/2.0/com/opsware/ws/
ejb/ServerWebService"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</input>
<output>
  <soap:body use="encoded"
  namespace="http://www.opsware.com/osapi/2.0/com/opsware/ws/
ejb/ServerWebService"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
. . .
<message name="getIDs" >
  <part name="serverFilter"
  xmlns:partns="java:com.opsware.ws.filter"
  type="partns:ServerFilter" />
</message>
<message name="getIDsResponse" >
  <part name="result"
  xmlns:partns="java:language_builtins.math"
  type="partns:ArrayOfBigDecimal" />
</message>
```

How the Chapter on the Complex Data Types Reference Documents the WSDL Files

The Complex Data Types chapter describes the elements of the `ServerFilter` type in a language-neutral manner. A SOAP complex data type is made up of other data types. The `ServerFilter` complex type, for example, contains a `string` type named `machineID` and a `dateTime` type called `lastRequestTimeStamp`. Basic types, such as `string`, `decimal`, and `dateTime`, are not complex types. The developer syntax docs specify the complex types in each language (Java, Python, Perl, C#). For example, the javadocs for the `ServerFilter` class specify the `machineID` and `lastRequestTimeStamp` elements by their corresponding getter and setter methods.

The following snippet of the `ServerWebService.wsdl` file shows some of the elements contained in the `ServerFilter` complex type:

```
<xsd:complexType    name="ServerFilter">
   <xsd:sequence>
    <xsd:element      name="machineID"
      maxOccurs="1"
      type="xsd:string"
      minOccurs="1"
      nillable="true">
    </xsd:element>
    <xsd:element      xmlns:tp="java:language_builtins.math"
      name="serverIDs"
      maxOccurs="1"
      type="tp:ArrayOfBigDecimal"
      minOccurs="1"
      nillable="true">
    </xsd:element>
    <xsd:element      name="lastRequestTimeStamp"
      maxOccurs="1"
      type="xsd:dateTime"
      minOccurs="1"
      nillable="true">
    </xsd:element>
. . .
 </xsd:sequence>
  </xsd:complexType>
```

Array and List Complex Types

The WSDL files define arrays as `ArrayOf<type>`. For example, the `serverIDs` element is an `ArrayOfBigDecimal`, which is defined in `ServerWebService.wsdl` as:

```
<xsd:complexType name="ArrayOfBigDecimal">
  <xsd:complexContent>
  <xsd:restriction xmlns:soapenc="http://schemas.xmlsoap.org/
soap/encoding/"
  base="soapenc:Array">
  <xsd:attribute xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  wsdl:arrayType="xsd:decimal[]"
  ref="soapenc:arrayType">
  </xsd:attribute>
  </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Because the WSDL files define every `ArrayOf<type>` as a `<type>[]`, the Complex Data Types chapter does not include entries for the array types. Whenever you see `ArrayOf<type>` in a reference chapter, you may assume that it is a `<type>[]`.

Also, the Complex Data Types chapter does not include entries for lists. A data type named `<type>List` embeds an array `<type>[]` and extends `ListBase`, which contains the `timeStamp` element.

Chapter 1: Overview

IN THIS CHAPTER

This chapter provides the following overview information for the Opsware System Web Services API 2.0:

Overview of the Opsware System Web Services API 2.0

The Opsware System Web Services API 2.0 exposes a Web Services interface to facilitate the integration of business and operations support systems with the Opsware platform. The Opsware System Web Services API 2.0 allows other IT systems – such as existing monitoring, trouble ticketing, billing, and virtualization technology – to exchange information with the Opsware platform.

Developers can use a Web Services-enabled development environment – such as Microsoft Visual Studio .NET, BEA WebLogic Workshop, or Perl scripts – to develop monitoring and reporting applications that invoke procedures through this interface.

The Opsware System Web Services API 2.0 is built on Web Services standard protocols, such as XML, SOAP (Simple Object Access Protocol), and WSDL (Web Services Definition Language), so that customers are not locked into proprietary protocols that make integration more complex and costly.

Overview of Features

The Opsware System Web Services API 2.0 provides the following features:

- A secure, programmable system architecture that enables programs to interact with the Opsware System.

- Product features written with programmability in mind to enable customers and systems integrators to extend features.
- A programmable environment that includes ten Web Services for major Opsware System functionality
- Sample code and developer syntax docs for each supported client language

With the Opsware System Web Services API 2.0, you can:

- Build new automation applications and extend Opsware System applications to improve IT productivity and comply with your own IT policies and procedures. You can use the Opsware System Web Services API 2.0 to drive external systems (such as network or storage automation systems) as well as to be driven by external systems.
- Integrate the Opsware System with other IT systems.
- Use the Opsware Model Repository as a central repository for storing and organizing critical IT information (operations, environment and asset information), making the repository the system of record.
- Automate a wide range of software and operating systems without having to wait for Opsware, Inc. to deliver out-of-the-box support for a particular technology.

What's New in Version 2.0

The Opsware System Web Services API 2.0 has the following new features:

- Splitting of the single Web Service into a set of Web Services

  In version 1.0, all operations were defined in a single Web Service. In version 2.0 the single Web Service was split into ten Web Services. The single WSDL that defined all Web Services and their operations was broken up into one WSDL per Web Service.

- A different context URI

  The Context URI of version 2.0 is different from version 1.0. In version 1.0, the API used the following URL:

```
https://<host>:<port>/opsw/ws?wsdl
```

  In version 2.0, the API uses the following URL:

```
https://<host>:<port>/osapi/2.0/com/opsware/ws/ejb/
<WebServiceName>
```

- Changes to the operation names in the Web Services

  See "Changes in Operation Names From Version 1.0 to 2.0" on page 12.

- New Command Session Web Service, with the following operations:
  - `get`: Returns the Command Engine `Session` object specified by the session ID.
  - `getDSEServerProgress`: Returns the progress status of a server for the Distributed Script Execution (DSE) session specified by the ID.
  - `getDSEServerResult`: Returns the result of a script running on a server for the DSE session specified by the ID.
  - `getDSESessionProgress`: Returns the progress status of the DSE session specified by the ID for all servers that are executing the script.
  - `getDSESessionResult`: Returns the result of the DSE session specified by the ID.
  - `getReconcileServerProgress`: Returns the progress of a reconcile session on a server specified by the ID.
  - `getReconcileServerResult`: Returns the result of the reconcile session specified by the ID.
  - `getReconcileServerResultPreview`: Returns the result of the reconcile preview session specified by the ID.
  - `getReconcileSessionProgress`: Returns the progress status of the reconcile session object specified by the sessionID parameter.
  - `getReconcileSessionResult`: Returns the result of the reconcile session specified by the ID.
  - `getReconcileSessionResultPreview`: Returns the result of the reconcile preview session specified by the ID.
  - `getStatus`: Returns the status of the session specified by the ID.
- New Distributed Script Web Service, with the following operations
  - `execute`: Starts a distributed script execution session.
- Additional operations in version 2.0
  - `move`: Used in the Node Web Service to move a node in the Software Tree.
  - `reconcile`: Used in the Server Web Service to start a reconcile session.

Web Services in this Version

Each Web Service in the Opsware System Web Services API 2.0 provides a set of operations that can be invoked through remote procedure calls. The signature of each operation includes arguments that take the form of either a unique identifier for an object or a filter that constrains the requested list of objects.

The Opsware System Web Services API 2.0 consists of the following Web Services:

- **Command Session Web Service** - Allows you to access progress and result information about two important Opsware Command Engine sessions: Distributed Script Execution (DSE) and reconcile. The progress information is available while the session is running, and the results are available after the session has completed execution.
- **Custom Attribute Web Service** - Used to apply special properties as name-value pairs to servers, nodes, customers, facilities, and OS definitions. You can get, set, or remove custom attributes.
- **Customer Web Service** - Enables you to create, remove, and retrieve customer accounts, the entities that allow you to create business units to provide management of Opsware System operations and configurations. You can also associate an Opsware customer with a facility.
- **Distributed Script Web Service** - Runs a script on one or more managed servers. The Distributed Script Execution (DSE) subsystem of Opsware supports these types of scripts: Unix/Linux shell, Windows batch (.BAT), and Windows Visual Basic (VBScript).
- **Facility Web Service** - Allows you to create, update, retrieve, or decommision an Opsware facility, which represents a collection of servers and server groups that are managed by a single Model Repository. Typically, a facility corresponds to a specific data center.
- **Node Web Service** - Used to create and delete a node in the Software Tree, the structure that organizes Opsware entities. The hierarchcy of the tree enables nodes such as software applications to inherit properties from other software application nodes. An Opsware entity is an object such as `Customer`, `Facility`, `Software Package`, and `Server`. To specify relationships, you can attach entities to nodes. For example, you can attach servers, software packages, customers, platforms to a Custom Application node.
- **Platform Web Service** - Retrieve information about a platform, which in the Opsware System refers to an operating system.

- **Security Web Service** - Enables you to perform security operations such as creating and removing Opsware security roles (groups) and users. You can also assign customer and facility permissions to roles.
- **Server Web Service** - Allows you to perform a variety of operations on an Opsware managed server, such as associating a server with a customer, checking the status of the Opsware Agent on the server, retrieving a list of changes for a server, and reconciling servers. (A reconcile operation installs or uninstalls software on a managed server.)
- **Software Web Service** - Retrieves information about the software packages that are defined in the Opsware System or installed on managed servers.

Version History

The following table lists the versions of the Opsware System Web Services API.

Table 1-1: Versions of the Opsware System Web Services API

OPSWARE SYSTEM WEB SERVICES VERSION	OPSWARE SYSTEM VERSION	MAJOR CHANGES
2.0	4.6, 4.7	Web Service split into multiple Web Services. Operation name changes from version 1.0 Web Services. Versioning scheme for backward compatibility. Addition of reconcile, distributed script execution, and move node operations.
1.0	4.5	Addition of the Security Web Service. Minor changes to the operation names in existing Web Services.

Table 1-1: Versions of the Opsware System Web Services API

OPSWARE SYSTEM WEB SERVICES VERSION	OPSWARE SYSTEM VERSION	MAJOR CHANGES
0.3, 0.2 , 0.1	4.0.1, 4.0.2, 4.0.3	Technology previews of the Opsware System Web Services API.

Architecture of the Opsware System Web Services API 2.0

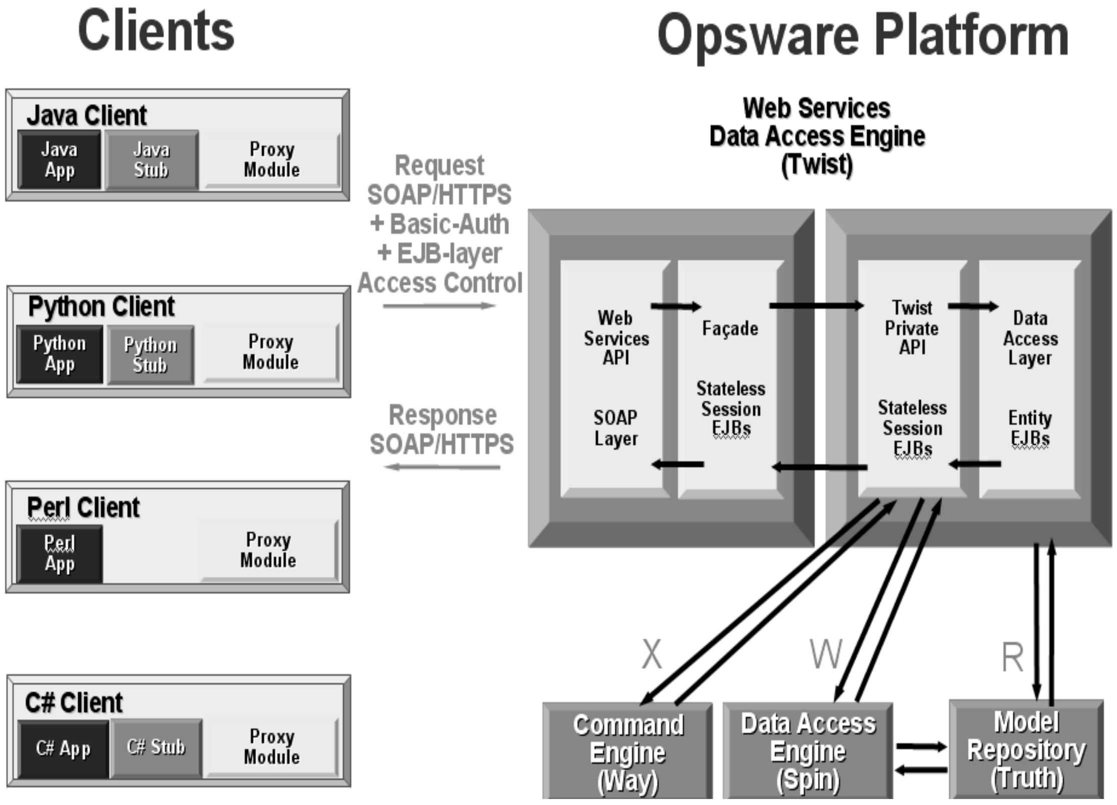
Figure 1-1 shows the overall architecture of the Opsware System Web Services API 2.0.

On the Opsware platform, the Opsware System Web Services API 2.0 is implemented by by two bundled components – the API itself and the Opsware Web Services Data Access Engine (twist). When you install the Opsware Command Center (a core component) on a machine, the Opsware Web Services Data Access Engine and the Opsware System Web Services API 2.0 are also installed.

The Opsware System Web Services API 2.0 is a wrapper around the Web Services Data Access Engine, providing a session facade around the internal APIs. These private APIs consist of session EJBs that use the entity EJBs to access data in the Opsware Model Repository (for write operations) and to access the Data Access Engine (for read operations).

Using SOAP over HTTPS, remote clients send requests to the Opsware platform and get a response back. The SOAP layer in the Opsware System Web Services API 2.0 receives the SOAP messages and invokes the corresponding methods in the façade layer, which is implemented as stateless session EJBs. The response sent to the client is in the form of a SOAP envelope, the body of which contains the data operation, a report on the outcome of the request, or a SOAP fault.

Figure 1-1: Architecture of the Opsware System Web Services API 2.0



Client Architecture

A Web Service client consists of three components:

- Client application

  A client application is written in one of these programming languages: Java, Python, Perl, or C#. A client application makes remote calls to the Opsware System by invoking the operations of the Opsware System Web Services API 2.0. For example, a client application might invoke operations on the Server Web Service to retrieve information about the servers managed by the Opsware System. A client application is created by a software developer, typically a system integrator, who is familiar with the Opsware System.

- Proxy module

A proxy module is a software library that implements SOAP messaging for a specific programming language. The Python ZSI proxy module, for example, converts native Python datatypes to and from SOAP messages. At runtime, the proxy module establishes the connection with the server. For a list of supported proxy modules, see Table 1-2 on page 9.

- Client stub

  The client stub defines the operation signatures and data types that are available in the Opsware System Web Services API 2.0. A client application invokes the Web Services operations on the client stub. Opsware, Inc. provides client stubs for each supported proxy module.

  Perl clients use dynamic proxies and do not require static stubs.

All three client-side components-- the client application, proxy module, and client stub-- must be compiled together.

WSDL Files and Web Services URLs

The Web Services are described in the Web Services Description Language (WSDL) format. Each WSDL file provides a complete description of the operations, arguments, return values, and complex data types of a specific Web Service. (Examples of complex data types are search filters, lists, arrays, and Opsware entities such as customers, facilities, and servers.) The `ServerWebService.wsdl` file, for example, defines the Opsware Server Web Service.

To access the Opsware System Web Services API 2.0, application clients specify the URL of the WSDL file for a particular service. The syntax of the WSDL file URL follows:

```
https://<occ-host>:<port>/osapi/2.0/com/opsware/ws/ejb/<web-
service>?wsdl
```

In the preceding URL, `<occ-host>` is the host on which the Opsware Command Center core component has been installed. (The Opsware System Web Services API 2.0 and the Web Services Data Access Engine are automatically installed on the same host as the Opsware Command Center.) The `<port>` is the port number ot the Web Services Data Access Engine. The default port number is 1031. The `<web-service>` is one of the following:

```
CommandSessionWebService
CustomAttributeWebService
CustomerWebService
DistributedScriptWebService
```

```
FacilityWebService
NodeWebService
PlatformWebService
SecurityWebService
ServerWebService
SoftwareWebService
```

For example, the following URL specifies the host `occ.c07.dev.opsware.com`, the default port number, and the WSDL location for the `ServerWebService`:

```
https://occ.c07.dev.opsware.com:1031/osapi/2.0/com/opsware/ws/
ejb/ServerWebService?wsdl
```

Supported Client Languages and Technologies

The Opsware System Web Services API 2.0 supports client applications written in Java, Python, Perl, or C#. The following table lists the specific versions of proxy modules that are supported for these client languages.

Table 1-2: Supported Proxy Modules for Client Applications

CLIENT LANGUAGE	SUPPORTED CLIENT PROXY MODULE	PROVIDER
Java	WebLogic 7.0 Web Services client	BEA Systems, Inc.
Python	Zolera SOAP Infrastructure (ZSI) 1.5.0	open source
Perl	SOAP::Lite 0.6	open source
C#	Microsoft .NET Framework SDK version 1.1	Microsoft Corp.

At the time of this writing, Java application clients have been tested on the Linux and Solaris operating systems. Although not fully tested, samples of Python and Perl clients have run successfully on Solaris and Linux, and samples of C# clients have run successfully on Windows 2000/2003.

You can run the client application on any server – you are *not* limited to running it on a server that is managed by the Opsware System (a server that has an Opsware Agent installed on it).

About the Developer Syntax Docs

For each supported client-application language (Java, Python, Perl, or C#), Opsware Inc. provides developer docs that define the language-specific syntax of the operations in each Web Service. These developer syntax docs were generated by tools that read the client stubs provided by Opsware Inc. Although they define the syntax for the operations, the developer docs do not provide semantic (descriptive) information. For descriptions of the operation calls, parameters, and return types, see the Web Services reference chapters in this guide.

Java Developer Docs (javadocs)

The Java docs are generated from the implementation of the Web Services using the javadoc tool. The Java docs describe:

- The interfaces of the stateless session EJBs implementing the Web Services
- The data structures for the input and output parameters of the operations in the Web Services interface

Only the public methods of the classes in the `com.opsware.ws.ejb` package are exposed as Web Service operations. These public methods have corresponding operation definitions in the WSDL files. The constructors and setters for Java beans in the other packages documented in the Java docs affect only the local objects in the client application, not the objects on the Opsware platform (server side). For example, the `Server.setAssetTag` method changes the `Server` object in the client application, but does not alter the corresponding object on the Opsware platform.

To view the methods of a specific Web Service:

1. In a browser window, go to the main index page for the Java docs of this release.
2. In the upper left pane, select `All Classes`.
3. In the lower left pane, select `<name>WebService`.

Python Developer Docs

The Python developer docs are generated from the Python client stub using Epydoc, a tool for generating API documentation for Python modules. When generating the Python client stub, the ZSI proxy module creates hundreds of classes in the client stub (four classes per operation plus additional classes for the objects). Epydoc generates an HTML file for each class.

To view the methods of a specific Web Service:

1. In a browser window, go to the main index page for the Python docs of this release.
2. In the upper left pane, select the module `<name>WebService_services`.
3. In the lower left pane, select the class `<name>WebServicePortSoapBindingSOAP`.

Perl Developer Docs

The Perl developer docs are generated from the Perl client stub using the Pdoc library.

To view the methods of a specific Web Service:

1. In a browser window, go to the main index page for the Perl modules docs of this release.
2. In the lower left pane, select `<name>WebService`.

   The methods (from the stub generated by SOAP::Lite) are listed in the section "Privates (from my definitions)."
3. Alternatively, at the top part of the page in the Toolbar section, select `Raw content`.
4. A page appears that displays the method signatures in a simpler format.

   The method parameters are included in the parameters field of each method. You can ignore the endpoint field because it is overwritten in the client application code with the appropriate endpoint location URL.

C# Developer Docs

The C# developer docs are generated from the C# client stub using NDoc, an extensible code documentation generation tool for .NET developers. You can view the C# developer docs as MSDN-online-style web pages (generated by Opsware Inc.). When generating the C# client stub, the Microsoft .NET Framework SDK creates two additional operations for each Web Service operation.

To view the methods of a specific Web Service:

1. In a browser window, go to the main index page for the C# docs of this release.
2. In the left pane, expand `<name>WebServiceProxy`.
3. In the left pane, expand `<name>WebService Class`.
4. In the left pane, select `<name>WebService Members`.
5. In the right pane, scroll down to the section Public Instance Methods.

Note each method name that starts with a lower case letter. These are the methods for the Web Service operations.

Ignore the following methods:

- Inherited methods. The C# stub contains methods inherited from classes in the .NET Framework SDK.
- Methods that start with `Begin` or `End`. For example, a `get` method will have a `BeginGet` and an `EndGet` method generated in the stub in addition to the actual `get` method.

6 To view the C# signature of a method, select the method name.

Changes in Operation Names From Version 1.0 to 2.0

Except for the operation names in the Security Web Service, most of the names of the Web Services operations that are available in version 1.0 changed in version 2.0. (The Command Session and the Distributed Script Execution Web Services are new in version 2.0.) This renaming in version 2.0 rationalizes the naming conventions and allows operation overloading across Web Services.

You can still access version 1.0 operations through the 1.0 URL of the Web Server. However, an API client application can invoke *only* operations within a single version of the API. You *cannot* mix calls to version 1.0 and version 2.0 operations in the same client. Therefore, if you are calling any operations within version 1.0 of the API, all operations called from that same client application need to access version 1.0. Contact your Opsware support representative for information about continued support for version 1.0.

Version 1.0 clients upgrading to version 2.0 must switch from the 1.0 to the 2.0 version of the client stub. Also, the client code must be modified and recompiled.

The following tables map the operation names from version 1.0 to 2.0:

- Custom Attribute Web Service Operation Names
- Customer Web Service Operation Names
- Facility Web Service Operation Names
- Node Web Service Operation Names

- Platform Web Service Operation Names
- Software Web Service Operation Names
- Server Web Service Operation Names

Table 1-3: Custom Attribute Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`setCustomAttributeValue`	`set`
`getCustomAttribute`	`get`
`getCustomAttributeList`	`getList`
`getCustomAttributeKeyList`	`getKeyList`
`getCustomAttributeValues`	`getValues`
`removeCustomAttributes`	`remove`

Table 1-4: Customer Web Service Operation Names

1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getCustomer`	`get`
`getCustomerList`	`getList`
`getCustomerIDList`	`getIDs`
`associateCustomerWithFacility`	`associateWithFacility`
`createCustomer`	`create`
`editCustomer`	`update`
`deleteCustomer`	`delete`

Table 1-5: Facility Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getFacility`	`get`
`getFacilityList`	`getList`
`getFacilityIDList`	`getIDs`
`createFacility`	`create`
`editFacility`	`update`
`decommisionFacility`	`decommission`

Table 1-6: Node Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`createNode`	`create`
`getNode`	`get`
`getNodeList`	`getList`
`getNodeIDList`	`getIDs`
`getParentNode`	`getParent`
`getChildrenNodeList`	`getChildList`
`getChildNode`	`getChild`
`getDescendantNode`	`getDescendant`
`attachServerToNode`	`attachServer`
`getAttachedServerList`	`getServerList`
`detachServerFromNode`	`detachServer`
`setSoftwarePackagesToNode`	`setSoftwarePackages`
`getAttachedSoftwarePackageList`	`getSoftwarePackageList`

Table 1-6: Node Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getAttachedSoftwarePackageList -ByInheritance`	`getInheritedSoftwarePackageList`
`getSoftwarePackageID - OverrideTupleList`	`getSoftwarePackageIDOverrideTupleList`
`detachSoftwarePackageFromNode`	`detachSoftwarePackage`
`setCustomersToNode`	`setCustomers`
`getAttachedCustomerList`	`getCustomerList`
`detachCustomersFromNode`	`detachCustomers`
`setPlatformsToNode`	`setPlatforms`
`getAttachedPlatformList`	`getPlatformList`
`detachPlatformsFromNode`	`detachPlatforms`
`getNodeListForServer`	`getNodeList`
`deleteNode`	`delete`
N/A	`move` (new in 2.0)

Table 1-7: Platform Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getPlatform`	`get`
`getPlatformList`	`getList`
`getPlatformIDList`	`getIDs`

Table 1-8: Software Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getSoftwarePackage`	`get`

Table 1-8: Software Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getSoftwarePackageList`	`getList`
`getSoftwarePackageIDList`	`getIDs`
`getInstalledSoftwarePackageList`	`getInstalledList`
`getAssociatedSoftwarePackageList`	`getAssociatedList`

Table 1-9: Server Web Service Operation Names

VERSION 1.0 OPERATION NAME	VERSION 2.0 OPERATION NAME
`getServer`	`get`
`getServerComponents`	`getComponents`
`getServerList`	`getList`
`getServerComponentsList`	`getComponentsList`
`getServerIDList`	`getIDs`
`getReachableServer`	`getReachableServer`
`getServerEventList`	`getEventList`
`getServerChangeLogList`	`getChangeLogList`
`checkServerReachability`	`getAgentStatus`
`associateServerWithCustomer`	`associateWithCustomer`
`updateServer`	`update`
`deleteServer`	`delete`
N/A	`reconcile` (new in 2.0)

Chapter 2: Setup

IN THIS CHAPTER

This chapter describes the following setup and usage information:

- Setup Requirements
- Authentication and Authorization
- Exception Handling
- Verbose Mode
- Logging
- About the Sample Applications

Setup Requirements

This section discusses the prerequisites for clients of the Opsware System Web Services API 2.0. The README files included with the sample applications also contain setup information.

Setup Requirements for All Client Languages

1. Verify that the version of the Opsware System Web Services API that your clients rely on is compatible with the version of the Opsware core to be accessed by the clients. To check the version numbers, see Table 1-1, "Versions of the Opsware System Web Services API," on page 5.

2. Make sure that the Opsware Web Services Data Access Engine is running and that the API is accessible by viewing one of the WSDL files in a browser. For an example URL, see "WSDL Files and Web Services URLs" on page 8.

3. Note the host and port in the URL you specified in the preceding step. Your client applications will specify these values in the WSDL URL when accessing the Opsware System Web Services API 2.0.

Java Client Setup

To set up your development environment for Java clients, you need the files listed in this section. Opsware Inc. provides a package (`j2ee.tar.gz`) that contains all of the required files and libraries.

Java Client Proxy Module

- WebLogic Server 7.0 Web Services client – `webserviceclient.jar`
- Available from Opsware Inc. or from the `${WEBLOGIC_HOME}/server/lib` directory on the host running the Opsware Web Services Data Access Engine.
- You can use the Axis Web Services client as the Java proxy module. The jar file is available from the Axis Web site (open source). To use the Axis proxy, you must create new Java client stubs and modify the Java sample client code accordingly.

CJava Client Stub

- Available from Opsware Inc.: `OpswareJavaSOAPStub.jar` (WebLogic-generated).
- If you choose not to use the Java client stub provided by Opsware Inc., you can generate your own client jar file that contains the client stubs, serializers, and deserializers. To generate your own client jar file, run the BEA-implemented `ant` task `clientgen`. See the BEA WebLogic Server documentation for information on `clientgen`.

Java Client Required Libraries

- Sun Microsystems JSSE library: `jsse-1_0_3_01-do.jar`
- Sun Microsystems JCE library: `jce1_2-do.jar`
- Sun Microsystems JNet library: `jnet.jar` (missing in JSSE 1.0.3)
- Sun Microsystems JCert library: `jcert.jar` (missing in JSSE 1.0.3)
- Opsware SSL initialization library: `common-1.2.0.jar` (available from Opsware Inc.)

Java Client Security Requirements

- JSSE Adapter class – `JSSEAdapter.java )`example available from Opsware Inc.)

  All clients accessing the Opsware System Web Services API 2.0 are required to use SSL to connect to the API. Only Java clients must set a JSSE adapter to use the SSL client.

Using the WebLogic Web Services SSL client to set the trusted server certificate will not work because it assumes that the default port is 443.

- Opsware trusted server certificate: `opsware-ca.crt`

  Java clients require a trusted server certificate to verify the server certificate presented to the client during the SSL handshake. Copy the `opsware-ca.crt` file from the Opsware Web Services Data Access Engine server directory to the client home directory. The Opsware Web Services Data Access Engine server directory is as follows:
  `/var/lc/crypto/twist`

  Make sure that the `CLASSPATH` used by the Java client includes the client home directory.

Python Client Setup

To set up your development environment for Python clients, you need the files listed in this section. Opsware Inc. provides a package (`python.tar.gz`) that contains the required client stub and types files. Each Web Service has a stub file and a types file.

Python Software Dependencies

Install the following software for your development environment:

- Python 2.3.3
- fpconst-0.6.0
- PyXML-0.8.3

Follow the installation instructions in the README files of these packages.

Python Client Proxy Module

- ZSI (Zolera SOAP Infrastructure) - open source
- Current version: 1.5.0

Python Client Stub

- You *must* use the stub provided by Opsware Inc. This ZSI-based stub was generated and tested with Python 2.3.3. (The wsdl2py stub generator script does not work properly with Python 2.2.) Opsware Inc. patched this stub because the float format for decimal caused certain operations to fail in the Opsware Web Services Data Access Engine.

ZSI generates the stub and types files from a WSDL file. The types file contains type definitions for the data types described in the WSDL. Opsware Inc. provides a stub file and a types files for each service. For example, the provided files for the Customer Web Service are as follows:

`CustomerWebService_services.py (`stub file)
`CustomerWebService_services_types.py` (types file)

Additional Python Client Requirements

- Set the `PYTHONPATH` environment variable to include the location path names for `fbconst-0.6.0`, `PyXML-0.8.3`, and `ZSI-1.5.0`. For example, in `csh` you might set `PYTHONPATH` as follows:

```
setenv PYTHONPATH /h/foo/wsapi/zsi/fpconst-0.6.0:/h/foo/wsapi/
soappy/PyXML-0.8.3:/h/foo/wsapi/zsi/ZSI-1.5.0
```

Known Issues for Python Clients

- The ZSI Decimal type uses a default float format for encoding decimal values in a SOAP request. WebLogic uses the string representation of the decimal to instantiate a BigDecimal object. When using the string representation of this BigDecimal to create an XML-RPC Long, the Long constructor fails. Workaround: Use the Opsware-provided patched stub for ZSI.
- The WebLogic-generated WSDL contains import statements as well as namespace-encoded import statements. The wsdl2py stub generator script fails in such situations. This problem was fixed by editing the WSDL file to remove the namespace-encoded import statements that are imported before invoking the wsdl2py stub generator. Workaround: Use the patched ZSI stub provided by Opsware Inc.

Using the SOAPpy proxy module to generate the Python client stub is not supported in this release. The SOAPpy proxy module does not correctly serialize complex objects, including the search filters of the Opsware System Web Services API 2.0.

Perl Client Setup

To set up your development environment for Perl clients, you will need the files listed in this section.

A Perl client does not require a static client stub to access the Opsware System Web Services API 2.0. The Perl proxy module uses a dynamic proxy to connect to the server. Therefore, you do not need to obtain Perl stub files from Opsware Inc.

Software Dependencies

Install the following software for your development environment:

- Crypt-SSLeay-0.51
- IO-Socket-SSL-0.95
- Net_SSLeay.pm-1.25
- HTML-Parser-3.35
- MIME-Base64-3.01
- URI-1.30
- libwww-perl-5.76

Follow the installation instructions in the README files of these packages.

Perl Client Proxy Module

- SOAP::Lite for Perl - open source
- Current version: SOAP-Lite-0.60a

Perl Client Stub

- A static stub is not required.

Known Issues

- User credentials must be encoded in the WSDL URL for Basic Authentication.
- Other mechanisms for specifying user credentials do not work.

C# Client Setup

To set up your development environment for C# clients, you must meet the following requirements. Opsware Inc. provides a package (`dotnet-csharp.zip`) that contains the required client stub file, `App.config` file, and certificate validation class. For instructions on setting up and running the sample client, see the README file included in `dotnet-csharp.zip`.

C# Software Dependencies

Install the following software for your development environment:

- Microsoft .NET Framework SDK version 1.1

C# Client Stub

- Opsware Inc. provides a stub file for each service, for example:
  `CustomerWebService.cs`

C# Client Proxy Module

- Microsoft .NET Framework SDK version 1.1.

Authentication and Authorization

For authentication, the Opsware System Web Services API 2.0 uses the BasicAuth mechanism, which establishes trust by validating a user name and password. To authorize access to specific operations, the API uses the WebLogic EJB security for roles and users.

Users of the Opsware System Web Services API 2.0 must be authenticated and authorized to invoke operations on the API. The [type product name here] provides three built-in user names and two built-in roles.

The roles and users of the Opsware System Web Services API 2.0 are independent of the roles, groups, and users known to the Opsware Command Center (OCC). The roles, groups, and users of the OCC are stored in the OCC Access and Authentication Directory and have privileges only within the OCC. The roles and users of Opsware System Web Services API 2.0 are not stored in the OCC Access and Authentication Directory and are not known by the OCC. Opsware Inc. inserts the roles and users of the API in the WebLogic-embedded LDAP. Note that in the next release, the OCC and Web Services users will be integrated into a single directory.

Built-in Users

The following table lists the user names that are built into the Opsware System Web Services API 2.0.

Table 2-10: Built-in Users

USER NAME	ASSIGNED ROLES	ACCESS TO MODEL REPOSITORY

`integration`	`wsapiUserRole`
	`wsapiReaderRole`
	`wsapiWriterRole`
		read and write

`wsapiReaderUser`	`wsapiUserRole` `wsapiReaderRole`	read only (accessor operations only)
`wsapiWriterUser`	`wsapiUserRole` `wsapiWriterRole`	write only

Passwords for Built-in Users

The default passwords are set when the Opsware System is installed and can be changed later with the WebLogic Administrative console.

Built-in Roles

The following table lists the roles that are built into the Opsware System Web Services API 2.0. A user can play one or more of these roles.

Table 2-11: Built-in Roles

ROLE NAME	DESCRIPTION
`wsapiReaderRole`	Users in this role can execute Web Service operations that read from the Model Repository.
`wsapiWriterRole`	Users in this role can execute Web Service operations that write to the Model Repository.
`wsapiUserRole`	This role is not for controling access at the operation level, but for authenticating the built-in users.

The built-in roles are defined in deployment descriptors within the Opsware Web Services Data Access Engine. You should not change the definitions of the built-in roles in the deployment descriptors.

Adding Users to the Opsware System Web Services API 2.0

If you want to use additional users with the Opsware System Web Services API 2.0 (other than the three built-in users), perform these tasks:

1. In the WebLogic Administrative console, add the users to the WebLogic-embedded LDAP.

   For instructions, see the BEA WebLogic Server documentation.

2. Assign the user either the `wsapiReaderRole`, the `wsapiWriterRole`, or both.

Users created in the WebLogic console are not persisted. Therefore, if the application server is restarted, these users will need to be recreated.

Because the WebLogic users are not integrated with the OCC users, only the Opsware System Web Services API 2.0 component recognizes new users.

HTTPS Access

Communication between client applications and the server component of the Opsware System Web Services API 2.0 is encrypted. The request and response SOAP messages (which implement the Web Service operation calls) are encrypted using SSL over HTTP (HTTPS).

Exception Handling

SOAP messages carry error information within the SOAP Fault element in the body of the SOAP message. The SOAP Fault element has four sub-elements:

- `faultcode`
- `faultstring`
- `faultactor`
- `detail`

Clients map the SOAP Fault element into data structures specific for each programming language. For the mapping in each language, see the tables in the sections that follow.

The Opsware System Web Services API 2.0 reports on runtime exceptions with a message in the `faultstring` sub-element of a SOAP Fault element. The `detail` sub-element contains a server-side stack trace.

The Opsware System Web Services API 2.0 does not implement error codes in this release. This restriction is due to a limitation in WebLogic 7.0, which is used for the Opsware Web Services Data Access Engine.

Java Client Exceptions

In the Opsware System Web Services API 2.0, Java clients can catch the `java.rmi.RemoteException`. The `SOAPFaultException` is wrapped in a `RemoteException`. See the code snippet that follows for an example on how to access the information in the `SOAPFaultException`. Due to the WebLogic 7.0 limitation mentioned previously, Java clients cannot catch the `SOAPFaultException` directly, nor can they access the `OpswareException`.

The following table shows the mapping of the SOAP Fault sub-elements to Java.

Table 2-12: Java Mapping for SOAP Faults

SOAP FAULT SUB-ELEMENT	JAVA MAPPING
`fautcode`	`SOAPFaultException.getFaultCode()`
`faultstring`	`SOAPFaultException.getFaultString()`
`faultfactor`	`SOAPFaultException.getFaultActor()`
`detail`	`SOAPFaultException.getDetail()`

For more information about this mapping, see the following URL:

http://java.sun.com/j2ee/1.4/docs/api/javax/xml/rpc/soap/SOAPFaultException.html

The following code snippet shows how a Java client would extract the `SOAPFaultException` from the `detail` field of the `RemoteException`:

```
...
catch ( RemoteException e ) {
   SOAPFaultException sfe = (SOAPFaultException) e.detail;
   System.out.println( e );
   System.out.println( sfe.getFaultCode() );
   System.out.println( sfe.getFaultString() );
   System.out.println( sfe.getFaultActor() );
   System.out.println( sfe.getDetail() );
   e.printStackTrace();
}
```

Python Client Exceptions

The following table shows the mapping of the SOAP Fault sub-elements to Python.

Table 2-13: Python Mapping for SOAP Faults

SOAP FAULT SUB-ELEMENT	PYTHON MAPPING
`fautcode`	`code`
`faultstring`	`string`
`faultfactor`	`actor`
`detail`	`detail`

For more information about this mapping, see the following URL:

http://pywebsvcs.sourceforge.net/zsi.html#SECTION0090000000000000000000

The following code snippet shows how a Python client would extract the SOAP Fault sub-elements:

```
try:
   response = wsProxy.method( request )
   result = response._result
except Exception, e:
   print "  Fault string: %s" % e.fault.string
   print "  Fault code: %s" % e.fault.code
   print "  Fault actor: %s" % e.fault.actor
   print "  Fault detail: %s" % e.fault.detail
```

Perl Client Exceptions

The following table shows the mapping of the SOAP Fault sub-elements to Perl.

Table 2-14: Perl Mapping for SOAP Faults

SOAP FAULT SUB-ELEMENT	PERL MAPPING
`fautcode`	`faultcode`
`faultstring`	`faultstring`
`faultfactor`	`faultactor`
`detail`	`faultdetail`

For more information about this mapping, see the following URL:

http://cpan.uwinnipeg.ca/htdocs/SOAP-Lite/SOAP/Lite.html#soap__fault

The following code snippet shows how a Perl client would extract the SOAP Fault sub-elements:

```
my $soap = SOAP::Lite .....;
my $som = $soap->method(@parameters);

 if ($som->fault) { # will be defined if Fault element
                    # is in the message
   print $som->faultdetail; # returns value of detail element as
                            # string or object
   $som->faultcode;   #
   $som->faultstring; # also available
   $som->faultactor;  #
   $som->result; # gives you access to result of call
                 # it could be any data structure
 }
```

C# Client Exceptions

The following table shows the mapping of the SOAP Fault sub-elements to C#.

Table 2-15: C# Mapping for SOAP Faults

SOAP FAULT SUB-ELEMENT	C# MAPPING
`fautcode`	`Code`

Table 2-15: C# Mapping for SOAP Faults

SOAP FAULT SUB-ELEMENT	C# MAPPING
`faultstring`	`Message`
`faultfactor`	`Actor`
`detail`	`Detail`

For more information about this mapping, see the following URL:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwebservicesprotocolssoapexceptionclasstopic.asp?frame=true

The following code snippet shows how a C# client would extract the SOAP Fault sub-elements:

```
catch (SoapException error) {
  // The exception details.
  Console.WriteLine("Fault Code Namespace" +
  error.Code.Namespace);
  Console.WriteLine("Fault Code Name" + error.Code.Name);
  Console.WriteLine("SOAP Actor that threw Exception" +
error.Actor);
  Console.WriteLine("Error Message" + error.Message);
  Console.WriteLine("Detail" +
HttpUtility.HtmlEncode(error.Detail.OuterXml));
}
```

Error Strings

The Opsware System Web Services API 2.0 returns the following error strings in the `faultstring` sub-element. The integers enclosed in curly braces are place holders for values that are filled in at run time.

```
Empty string argument.
Error when trying to retrieve list of {0} IDs.
Invalid agent status found for server with ID ''{0}''.
Invalid {0} filter argument.
Error when trying to create list of {0} objects.
Error when trying to retrieve list of {0} objects.
No matching {0} object found.
{0} ID list in filter argument not empty.
Null change log end time argument.
Null {0} filter argument.
Null {0} ID argument.
Null change log start time argument.
Null string argument.
Null last request time stamp in {0} filter argument.
No {0} object found with ID ''{1}''.
No {0} object found with IP address ''{1}''.
More than 1 {0} object found with IP address ''{1}''.
Error trying to retrieve too many {0} objects.
Last request time stamp in {0} filter argument is past current
time.
Last request time stamp in {0} filter argument is prior to
current cache sliding window start time.
Internal error when connecting to Web Services Data Access
Engine: {0}
```

Verbose Mode

You can invoke Web Services in verbose mode to view the SOAP envelopes as they are transmitted in the request and response messages. The following sections show how to change the sample client applications to invoke the Web Services in verbose mode.

Java Clients in Verbose Mode

In the `java` command that runs the client application, add the following property:

```
-Dweblogic.webservice.verbose=true
```

For example, the `java` command in the sample `wsapi.sh` file would be modified as follows:

```
${JAVA_HOME}/jre/bin/java -classpath ${MYCLASSPATH}
-Dweblogic.webservice.verbose=true
com.opsware.ws.demo.WebServicesDemo
```

Perl Clients in Verbose Mode

In the following code snippet from the sample client (`wsapi.pl`), uncomment the last line and add a comma after the closing curly bracket on the previous line:

```
use SOAP::Lite +autodispatch =>
   uri => 'http://www.opsware.com/opsw/OpswareWebService',
   proxy => 'https://
integration:integration@occ.c07.dev.opsware.com:1031/opsw/ws',
   on_fault => sub { my ($soap, $res) = @_;
       die ref $res ? $res->faultdetail : $soap->transport-
>status, "\n";
   }
   # trace => 'debug' # uncomment to trace SOAP messages
```

Python Clients in Verbose Mode

In the following code snippet from the sample client (`wsapi.py`), uncomment the last line and replace the closing curly bracket on the previous line (containing `url`) with a comma:

```
kw = {'auth': (AUTH.httpbasic, username, password),
     'host': host,
     'port': port,
     'ssl': 1,
     'url': url}
     # 'tracefile': sys.stdout} # uncomment to trace SOAP 
messages
```

Logging

The Opsware System Web Services API 2.0 writes entries for completed transactions in the two Web Services Data Access Engine log files:

- `/var/lc/twist/twist.log` – WebLogic-specific error or informational messages.
- `var/lc/twist/stdout.log` – Debug output and logging of every exception that is generated by the server. This log file does not conform to a specific format.

The entries from the Opsware System Web Services API 2.0 in `stdout.log` include the following. The integers enclosed in curly braces are place holders that are filled in at run time.

```
Method {0} invoked.
Returning {0} object with ID ''{1}''.
Returning {0} object with IP address ''{1}''.
Agent status for server with ID ''{1}'' is now ''{0}''.
About to create a list with {1,number,integer} {0} objects.
About to create a list with {1,number,integer} {0} IDs.
Time spent preparing results: {0,number,integer} msec.
Caught {0}: {1}
Servers removed:{0,number,integer}.
Servers modified: {0,number,integer}.
Servers added: {0,number,integer}.
Sliding window start time: {0}.
Current time: {0}.
Last request time stamp: {0}.
{0} object with ID ''{1}'' updated.
{0} object with ID ''{1}'' deleted.
```

```
Server with ID ''{0}'' moved to customer with ID ''{1}''.
```

About the Sample Applications

For each supported client language (Java, Python, Perl, or C#), Opsware Inc. provides sample applications that demonstrate the Opsware System Web Services API 2.0. All the samples connect to the same Web Services and invoke the same operations. Generally, the samples illustrate how to use get operations, including the use of search filters.

For detailed information about how to modify and run the samples in your environment, see the README files included in the sample application packages. See also, “Setup Requirements” on page 17.

Your client application should be compiled with the client stub, which is provided by Opsware Inc. The client stub that you obtain from Opsware Inc. should not be modified.

Sample Application Packages

The following table lists the contents of the sample application packages provided by Opsware Inc.

Table 2-16:

CLIENT LANGUAGE	PACKAGE FILE NAME	CONTENTS OF PACKAGE

Java	`j2ee.tar.gz`	`WebServicesDemo.java`
		`JSSEAdapter.java`
		`Makefile`
		Opsware security certificate
		`wsapi.sh` (script to run client)
		client stubs
		README

Python	`python.tar.gz`	`wsapi.py` client stubs README
Perl	N/A	`wsapi.pl`

Table 2-16:

CLIENT LANGUAGE	PACKAGE FILE NAME	CONTENTS OF PACKAGE

C#	`dotnet-csharp.zip`	`WebServicesDemo.cs`
		`MyCertificateValidation.cs`
		`App.config`
		client stubs
		README

Steps Performed by the Sample Client Applications

Each sample client application performs the following steps:

1. Initializes an SSL client.

   All clients must use SSL to connect to the Opsware System Web Services API 2.0. Java clients must validate the trusted server certificate of the Opsware System by setting a JSSE adapter to use the SSL client.

2. Binds to the Server and Custom Attribute Web Services by providing the URLs of the services.

   In the sample clients, the URLs are constructed by concatenating string values that are either hard coded in the source code, set in a shell script, or specified in an application configuration file (C#). You must modify the URL's host name and you may need to modify the port number. For instructions on modifying the URLs, see the README files included with the sample clients. See also, "WSDL Files and Web Services URLs" on page 8.

   The client application binds to a Web Service only once, and uses the port binding to invoke operations for that Web Service

3. Invokes the following Server Web Service operations and prints the results:

   ```
   get
   getAgentStatus
   getComponents
   getIDs
   getList
   ```

   The client passes the `serverFilter` argument to the `getIDs` and `getList` operations.

For more information about these operations, see "Server Web Service Operations Reference" on page 125. For more information about the `serverFilter`, see "Complex Data Types Reference" on page 149.

4 Invokes the following Custom Attribute Web Service operations and prints the results:

```
set
get
getKeyList
getList
remove
```

The client passes the `customAttributeFilter` argument to the `set` and `remove` operations.

For more information about these operations, see "Custom Attribute Web Service Operations Reference" on page 45. For more information about the `customAttributeFilter`, see "Complex Data Types Reference" on page 149.

Chapter 3: Command Session Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

The Command Session Web Service allows you to access progress and result information about two important Command Engine sessions: Distributed Script Execution (DSE) and reconcile. The progress information is available while the session is running. The results are available after the session has finished running.

The Opsware Command Engine runs distributed programs across many servers by interacting with Opsware Agents installed on the servers. Command Engine scripts are written in Python and run on the Command Engine server.

DSE provides the ability to configure scripts to run simultaneously across multiple Unix or Windows servers, and monitor script execution on each server. After a script runs, job- and server-specific execution results are available for review. To run a script, invoke the `DistributedScript.Execute` operation.

The reconcile function updated the actual software configuration of a server based on the specified configuration stored in the Model Repository ("The Truth"). During a reconcile, sofwtare, applications, and patches may be installed or uninstalled on one or more servers. To launch a reconcile, invoke the `Server.reconcile` operation.

get

Gets the specified `CommandSession` object.

The results of a Command Engine session provide such information as the name of the agent through which the session was run, the start and end date of the session, the person who initiated the session, session descriptions, the results digest, and so on.

Result

SOAP Data Type: `CommandSession`

Description: A `CommandSession` object, which contains information about a specific Command Engine session.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	The ID of the `CommandSession` object to retrieve.

getDSEServerProgress

Gets progress information of a DSE session for an individual server, including the number of completed steps, error details, script status, and so on.

Result

SOAP Data Type: `DSEServerProgress`

Description: Progress information about a current DSE session for an individual server.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the DSE session from which to retrieve server progress information.
`serverID`	`decimal`	ID of the server from which to retrieve DSE session progress information.

getDSEServerResult

Gets the results of a DSE session for a specified server, including the script execution, errors (if there are any) and status).

Result

SOAP Data Type: `DSEServerResult`

Description: Results information about a DSE session for an individual server.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the DSE session results to retrieve for an individual server.
`serverID`	`decimal`	ID of the server from which to retrieve DSE session results.

getDSESessionProgress

Gets the progress status of a current DSE session, including the total scripts run for each server as the scripts executes.

Result

SOAP Data Type: `DSESessionProgress`

Description: Progress information about a DSE session.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the DSE session from which to retrieve progress information.

getDSESessionResult

Gets the results of a DSE session, such the status and results of each server affected by the session, any errors that were encountered, and so on.

Result

SOAP Data Type: `DSESessionResult`

Description: Results information about a DSE session.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the DSE session from which to retrieve results.

getReconcileServerProgress

Returns progress status of a reconcile session for an individual server.

Result

SOAP Data Type: `ReconcileServerProgress`

Description: Progress information for a current reconcile session on a server.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the reconcile session from which to retrieve progress information.
`serverID`	`decimal`	ID of the server from which to retrieve reconcile progress information.

getReconcileServerResult

Gets the results information of a reconcile session for an individual server, such as details about the packages installed or uninstalled on the server.

Result

SOAP Data Type: `ReconcileServerResult`

Description: Results information about script execution for an individual server.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the reconcile session from which to get results for a specified server.
`serverID`	`decimal`	ID of the server from which to get reconcile results information.

getReconcileServerResultPreview

Gets the results of a reconcile preview session for an individual server. The preview reconcile, which is run before each full reconcile, allows you to see exactly what will happen to the server as a result of the software requested to be installed or uninstalled. A preview shows what will occur if the reconcile is initiated, what software will be installed and uninstalled, details about the software packages, and so on.

Result

SOAP Data Type: `ReconcileServerResultPreview`

Description: Preview information about a reconcile session.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the reconcile preview session to retrieve.
`serverID`	`decimal`	ID of the server from which to retrieve reconcile preview information.

getReconcileSessionProgress

Gets the progress status of the reconcile session, including how many servers have been reconciled and progress information for each server affected by the session.

Result

SOAP Data Type: `ReconcileSessionProgress`

Description: Details about the progress of a current reconcile session.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the reconcile session from which to retrieve progress information.

getReconcileSessionResult

Gets the results of a reconcile session, including how many servers successfully completed the reconcile, details about the software packages installed, and so on.

Result

SOAP Data Type: `ReconcileSessionResult`

Description: Result information from a reconcile session.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the Reconcile session to retrieve results of.

getReconcileSessionResultPreview

Gets the results of a reconcile session preview. The preview reconcile, which is run before each full reconcile, allows you to see exactly what will happen to the server as a result of the software requested to be installed or uninstalled.

Preview reconcile shows what packages will be installed and what packages will be removed. If a package is removed or installed as a result of another package being installed, this information is noted as a “side effect.”

Preview reconcile also indicates which package installs or uninstalls require a reboot.

Result

SOAP Data Type: `ReconcileServerResultPreview`

Description: Reconcile preview results information.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	ID of the reconcile preview session results to retrieve.

getStatus

Gets the status of the specified `CommandSession` object.

Result

SOAP Data Type: `string`

Description: Text message indicating the status of the `CommandSession` object.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionID`	`decimal`	The ID of the `CommandSession` object from which to retrieve status information.

Chapter 4: Custom Attribute Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

- get
- getKeyList
- getList
- getValues
- remove
- set

The custom attribute function is used to apply special properties to servers, nodes, customers, facilities, and OS definitions. You can set custom attributes which include setting miscellaneous parameters and named data values.

The named values in a custom attribute provide parameters to the Opsware System, for example, to customize displays or provide settings to use during installation or configuration of packaged software in the operational environment.

get

Returns a custom attribute specified by the object type, object ID, and key. All three arguments are required.

Result

SOAP Data Type: `CustomAttribute`

Description: The object that represents the custom attribute.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION

`objectType`	`string`	The type of Opsware entity the custom attribute has been assigned to. The `objectType` may be one of the following:
		`CUSTOMER`
		`FACILITY`
		`NODE`
		`SERVER`

`objectID`	`decimal`	The unique ID of the Opsware entity the custom attribute has been assigned to. For example, if the `objectType` is `CUSTOMER`, then the `objectID` is the same as the `customerID`.
`key`	`string`	The name of the custom attribute to be returned.

getKeyList

Returns custom attribute names (keys) by object type and object ID.

Result

SOAP Data Type: `CustomAttributeKeyList`

Description: An array of custom attribute object names.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION

`objectType`	`string`	The type of Opsware entity the custom attribute has been assigned to. The `objectType` may be one of the following:
		`CUSTOMER`
		`FACILITY`
		`NODE`
		`SERVER`

`objectID`	`decimal`	The unique ID of the Opsware entity the custom attribute has been assigned to. For example, if the `objectType` is `CUSTOMER`, then the `objectID` is the same as the `customerID`.

getList

Returns custom attributes by object type and object ID.

Result

SOAP Data Type: `CustomAttributeList`

Description: An array of custom attribute objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION

`objectType`	`string`	The type of Opsware entity the custom attribute has been assigned to. The `objectType` may be one of the following:
		`CUSTOMER`
		`FACILITY`
		`NODE`
		`SERVER`

`objectID`	`decimal`	The unique ID of the Opsware entity the custom attribute has been assigned to. For example, if the `objectType` is `CUSTOMER`, then the `objectID` is the same as the `customerID`.

getValues

Returns the custom attribute values that match the criteria specified by the `customAttributeFilter` argument. All three elements in the `customAttributeFilter` must be specified.

Result

SOAP Data Type: `CustomAttributeList`

Description: An array of custom attribute object values.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customAttributeFilter`	`CustomAttributeFilter`	The object that specifies the search criteria for this operation.

remove

Deletes custom attributes. The criteria to delete custom attributes are specified by the `customAttributeFilter` argument. All three elements in the `customAttributeFilter` are required.

Result

SOAP Data Type: `string`

`Description: The string OK.`

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customAttributeFilter`	`CustomAttributeFilter`	The object that specifies the search criteria for this operation

set

Sets the value of a custom attribute. All three elements in the `customAttributeFilter` argument are required. The `customAttributeFilter.customAttributeKeys` must contain a single array element. (In other words, just one key may be specified.)

Result

SOAP Data Type: `string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customAttributeFilter`	`CustomAttributeFilter`	The object that specifies the search criteria for this operation.
`customAttributeValue`	`String`	The value to which the custom attribute will be set.

Chapter 5: Customer Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

In the Opsware System, customer accounts allows to create business units to provide management of Opsware System operations and configurations.

By default, the Opsware System is shipped with the following two customers:

- Customer Independent - A global customer in the Opsware System. Resources (applications, patches, and templates) that are associated with "Customer Independent" can be installed on any managed server, no matter what customer it is associated with.
- Not assigned - The servers are not associated with a customer. You can install applications, patches, or templates that are Customer Independent on Not Assigned servers. However, you cannot install or use any resources associated with a customer on a server that is not assigned to a customer.

associateWithFacility

Associates a customer with a facility. The appropriate customers should be associated with each new facility so that servers managed at that facility are associated with the correct customers.

Result

SOAP Data Type: `string`

Description: The result of this operation.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerID`	`decimal`	The ID of the customer to be associated with the facility.
`facilityID`	`decimal`	The ID of the facility to which the customer will be associated.

create

Creates a new customer. After you create a customer, you can define the custom attributes for that customer.

Result

SOAP Data Type: `decimal`

Description: The ID (primary key) of the new customer.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerName`	`string`	The name of the new customer in the Opsware System. The value must be in uppercase, less than 25 characters, and cannot contain spaces.
`businessCustomerID`	`decimal`	The business ID of the new customer.
`displayName`	`string`	The display name of the new customer in the Opsware Command Center. The value must contain less than 50 characters and cannot include any special charatcters.
`authorizationDomain`	`string`	The authorization domain of the new customer. The value must be uppercase, less than 50 characters, and in the domain name format. This value is usually the same as the domain name.

delete

Deletes a customer. Deleting a customer removes the customer information from the Opsware Access & Authentication Directory, and moves deactivated servers assigned to the customer to the “Not Assigned” customer account or deletes the data about the servers from the Model Repository database.

You can only delete a customer from the Opsware Command Center when the following conditions are true for the customer:

- No Nodes are attached to the customer account
- The customer account does not own any software packages; the packages uploaded for the customer must be deleted or deprecated
- All servers assigned to the customer are deactivated
- No IP range groups are created for the customer
- No IP ranges are created for the customer
- No server groups are created for the customer.

Result

SOAP Data Type: `string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTIONS
`customerID`	`decimal`	The ID of the customer to be deleted.

get

Returns a customer specified by the ID.

Result

SOAP Data Type: `Customer`

Description: The object that represents the customer.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTIONS
`customerID`	`decimal`	The ID of the customer to be returned.

getIDs

Returns an array of customer IDs that matches the criteria specified by the `customerFilter` argument. To search for all acitve customers, specify a null `customerFilter`. To search for a customer by name, specify the `customerFilter.customerName`. To search for all customers associated with a facility, specify the `customerFilter.facilityID`.

This operation searches for customers with the following algorithm:

```
if customerFilter is null
return an array with the IDs of all active customers
else if customerFilter.customerIDs is not empty
throw an exception
else if customer.customerName is not null
return a single-element array with the ID of the
corresponding customer
else if customer.facilityID is not null
return an array with the IDs of all customers associated with
the specified facility
else
throw an exception
```

Result

SOAP Data Type: `decimal[]`

Description: An array of customer IDs.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerFilter`	`CustomerFilter`	The object that specifies the search criteria for this operation.

getList

Returns a list of customers that matches the criteria specified by the `customerFilter` argument.

The search performed by this operation follows the same criteria specified by the customerFilter argument documented for getIDs, except for the customerFilter.customerIDs array. In the getList operation, to search for a customer by ID, specify one or more values in the customerFilter.customerIDs array.

Result

SOAP Data Type: `Customer List`

Description: An object that represents multiple customers.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerFilter`	`CustomerFilter`	The object that specifies the search criteria for this operation.

update

Edits a customer.

Result

SOAP Data Type: `string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerID`	`decimal`	The ID of the customer to be edited. The ID itself cannot be changed.
`customerName`	`string`	The name of the customer to be edited in the Opsware system.The value must be in uppercase, less than 25 characters, and cannot contain spaces.
`businessCustomerID`	`decimal`	The business ID of the customer to be edited. (is it same as customerID? is it created automatically)
`displayName`	`string`	The display name of the customer to be edited. This name appears in the Opsware Command Center.

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`status`	`string`	The status of the customer to be edited. (The only status in OCC is ACTIVE. Is this the default status? can a user assign the status?)

Chapter 6: Distributed Script Web Service Reference

I N T H I S C H A P T E R

This chapter describes the syntax of the following SOAP operations:

- execute

execute

Runs a script on one or more managed servers. The Distributed Script Execution (DSE) subsystem of Opsware supports these types of scripts: Unix/Linux shell, Windows batch (.BAT), and Windows Visual Basic (VBScript).

The Opsware Command Engine handles the entry of scripts into the Opsware Model Repository (the script storage location in the Opsware System) and the versioning of stored scripts. During script execution on the servers, the Command Engine runs a script that issues an execution command to the Opsware Agent on each server. Each Opsware Agent handles script execution and sends execution results to the Command Engine, which enters the execution results data into the Model Repository.

To check the status of a script execution, invoke the `CommandSession.getDSESessionProgress` operation. To check the results of the script execution, invoke `CommandSession.getDSESessionResult`.

For tips and information on required permissions, see the online help of the Opsware Command Center or the Script Execution Subsystem chapter of the User's Guide.

Result

SOAP Data Type: `decimal`

Description: The ID of the Command Engine session handling the script execution.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`dseArgument`	`DistributedScriptArgument`	The complex type that specifies the arguments for running the distributed script. The arguments include the customer IDs associated with the servers and the parameters such as the server IDs and the script text.

Chapter 7: Facility Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

- create
- decommission
- get
- getIDs
- getList
- update

In the Opsware System, a facility refers to the collection of servers and server groups that are managed by a single Model Repository, the database that stores information about the managed environment. Typically, a facility corresponds to a specific data center. Users can manage servers in any facility from the Opsware Command Center in any facility. When a user updates data in a facility, the Model Repository for that facility is synchronized with the Model Repository located in all remote facilities. In the Opsware Command Center, a facility is identified by a customer-assigned facility name and an Opsware-assigned facility ID.

create

Creates a new facility.

Result

SOAP Data Type: `decimal`

Description: The ID (primary key) of the new facility.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityName`	`string`	The short name of the new facility. The value must be in uppercase and cannot contain spaces.
`displayName`	`string`	The name of the new facility displayed in the Opsware Command Center.

decommission

Removes a facility from use. In the Opsware Command Center, this action is called “deactivate.”

Decommissioning a facility stops Model Repository updates from propagating to the deactivated facility and stops the Multimaster Tools from functioning for that facility. Be sure to deactivate facilities with care because this task cannot be undone.

When you deactivate a facility, the facility is still listed in the Opsware Command Center; however, it is marked with the letter D. The facility short name cannot be re-used, even if the facility is deactivated.

Result

SOAP Data Type: `string`

Description: The result of the operation.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityID`	`string`	The ID of the facility to be decommissioned (deactivated).

get

Returns the facility specified by the ID

Result

SOAP Data Type: `Facility`

Description: The object that represents the facility.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityID`	`decimal`	The ID of the facility to retrieve.

getIDs

Returns an array of facility IDs that matches the criteria specified by the `facilityFilter` argument. To get all facilities, specify a null `facilityFilter`. To search for all facilities associated with a customer, specify the `facilityFilter.customerID`. If `facilityFilter.facilityIDs` is not null, this operation throws an exception.

Result

SOAP Data Type: `decimal[]`

Description: An array of facility IDs

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityFilter`	`FacilityFilter`	The object that specifies the search criteria for this operation.

getList

Returns a list of facilities that matches the criteria specified by the `facilityFilter` argument. This operation retrieves facilities in one of the following ways: To search for all facilities, specify a null `facilityFilter`. To search for active facilities by ID, specify `facilityFilter.facilityIDs`. To search for all facilities associated with a customer, specify the `facilityFilter.customerID`.

Result

SOAP Data Type: `FacilityList`

Description: An object that represents multiple facilities.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityFilter`	`FacilityFilter`	The object that specifies the search criteria for this operation.

update

Edits facility's attributes.

Result

SOAP Data Type: `string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityID`	`decimal`	The ID of the facility to be updated. The ID itself cannot be changed.
`facilityName`	`string`	The short name of the facility. The value must be in uppercase and cannot contain spaces.
`displayName`	`string`	The name of the facility displayed by the Opsware Command Center.
`status`	`string`	The facility's status, either `ACTIVE` or `DECOMMISSIONED`.

Chapter 8: Node Web Service Operation Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

In the Opsware System, a node is an element in a tree that organizes Opsware entities. The hierarchical structure of the tree enables nodes, such as software applications, to inherit properties from other software application nodes. An Opsware entity is an object such as `Customer`, `Facility`, `Software Package`, and `Server`. Behind the scenes, the Opsware System stores these entities in the Model Repository database. To specify relationships, you can attach entities to nodes. For example, you can attach servers, software packages, customers, platforms to a Custom Application node.

The node tree is made up of subtrees called stacks. Some of these stacks are for internal use by the Opsware System, and others, such as the System Utilities stack, can be manipulated by end users in the Opsware Command Center.

attachServer

Attaches a server to a node.

Result

SOAP Data Type: `string`

Description: Text message describing the results of attaching a server to a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	The ID of the node to be associated with the server.
`serverID`	`decimal`	The ID of the server to be associated with the node.

create

Creates a new node in any of the any of the six Application stacks, which include the Application Servers, DatabaseServer, OS Extras, Other Applications, System Utilities, and WebServer stacks. The position in the tree where the node is created is determined by the `parentNodeID`, which is the first parameter of the create operation. The created node will become a child of the specified parent node.

Result

SOAP Data Type: `decimal`

Description: ID of the new node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`parentNodeID`	`decimal`	The parent node ID under which you want to create a new node.
`shortName`	`string`	A short name for the new node. The value must be all upper case and contain no spaces.
`description`	`string`	A brief description of the new node.
`notes`	`string`	Notes that describe the node you are creating.
`allowDevice`	`boolean`	Specify if you want to allow devices (servers) to be attached to the new node.
`blockedNode`	`boolean`	Specify if you want the node to be locked. If a node is locked, it cannot be moved or edited except by a user who has the appropriate privileges.

delete

Deletes a node from the tree.

Result

SOAP Data Type: `string`

Description: Text message describing the results of deleting a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node to delete.

detachCustomer

Detaches a list of customers from a node.

Result

SOAP Data Type: `string`

Description: Text message describing the results of detaching a list of customers from a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node to detach from customers.
`customerIDs`	`decimal[]`	An array of customer IDs to detach from a node.

detatchPlatforms

Detaches a list of platforms (operating systems) from a node.

Result

SOAP Data Type: `string`

Description: A text message describing the results of detaching a list of platforms from a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node to detach from the platforms.
`platformIDs`	`decimal[]`	An array of platform IDs to detach from a node.

detachServer

Detaches a server from a node.

Result

SOAP Data Type: `string`

Description: Text message describing the results of detaching a server from a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	The ID of the node to detach servers.
`serverID`	`decimal`	The ID of the server to detach from the node.

detatchSoftwarePackage

Detaches a software package from a node.

Result

SOAP Data Type: `string`

Description: A text message describing the results of detaching a software package from a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node to detach from the software package.
`softwarePackageID`	`decimal`	ID of the software package to detach from the node.

get

Gets a `Node` object specified by the node ID. A `Node` object contains information such as its ID (unique identifier), short name, and parent ID.

Result

SOAP Data Type: `Node`

Description: A `Node` object.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node to retrieve.

getChild

Returns the `Node` object for the immediate descendant (specified by short name) of the node specified by ID. A `Node` object contains information such as its ID (unique identifier), short name, and parent ID.

Result

SOAP Data Type: `Node`

Description: A `Node` object.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the parent node whose child you want to get.
`childShortName`	`string`	The internal child node name used by the Opsware System. The value must be all upper case and contain no spaces.

getChildList

Gets a list of children node objects (the immediate descendents) that belong to a specified parent node.

Result

SOAP Data Type: `NodeList`

Description: A list of `Node` objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the parent node whose children you want to get.

getCustomerList

Gets a list of `Customer` objects that are associated with a node. A `Customer` object represents a customer entity in the Opsware System, and consists of such attributes as a customer ID, customer name, display name, status, authorization domain, and so on.

Result

SOAP Data Type: `CustomerList`

Description: A list of `Customer` objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node whose customer list you want to get.

getDescendant

Returns the `Node` object for the descendant (specified by short name path) of the node specified by node ID.

Result

SOAP Data Type: `Node` object.

Description: A `Node` object contains information such as its ID (unique identifier), short name, and parent ID.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node whose descendent you want to get.
`descendantShortNamePath`	`java.lang.String[]`	An array of strings representing the shortpath name of the descendent nodes you are getting.

getIDs

Gets a list of node IDs. Each node in the Opsware system contains a unique ID to distinguish it from all other nodes.

Result

SOAP Data Type: `decimal[]`

Description: An array of node IDs.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeFilter`	`nodeFilter`	Determines the search results of the `getIDs` operation.

getInheritedSoftwarePackageList

Gets a list of `Software` objects (software packages) that a node inherits from its parents and ancestors in the tree.

Result

SOAP Data Type: `SoftwarePackageList`

Description: A list of `Software Package` objects, which represent all packages that the specified node inherits from its parents and ancestors in the tree.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node from which you want to get a list of attached software packages.

getList

Gets a list of `Node` objects.

Result

SOAP Data Type: `NodeList`

Description: A list of `Node` objects. A `Node` object contains information such as its ID (unique identifier), short name, and parent ID.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeFilter`	`NodeFilter`	Determines the search results of the `getList` operation.

getNodeList

Gets a list of `Node` objects that are associated with a specified server.

Result

SOAP Data Type: `NodeList`

Description: A list of `Node` objects. A `Node` object contains information such as its ID (unique identifier), short name, and parent ID.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	ID of the server with the `Node` objects retrieved.

getParent

Returns the parent `Node` object for the node specified by ID.

Result

SOAP Data Type: `Node`

Description: A `Node` object. A `Node` object contains information such as its ID (unique identifier), short name, and parent ID.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node whose parent you want to get.

getPlatformList

Gets a list of `Platform` objects of a specified node. A `Platform` object represents the definition of an operating system in the Opsware System. A platform is the name and version of an operating system (for example, Red Hat Linux 4.2). Each `Platform` object contains varied elements such as what type of hardware is associated with the OS, OS version, a platform ID, platform name, and so on.

Result

SOAP Data Type: `PlatformList`

Description: A list of `Platform` objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node from which you want to get a list of platform objects.

getServerList

Gets a list of `Server` objects associated with a node. A `Server` object represents a server machine managed by the Opsware System, and contains such elements as a machine ID, OS version, serial number, server ID, and so on.

Result

SOAP Data Type: `ServerList`

Description: A list of `Server` objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node from which you want to get a list of platform objects.

getSoftwarePackageIDOverrideTupleList

Gets a list of `SoftwarePackageIDOverrideTuple` objects. This list indicates the installation order of inherited software packages.

Result

SOAP Data Type: `SoftwarePackageIDOverrideTupleList`

Description: A tuple in an ordered list that indicates the installation order of a node's inherited software packages.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node from which you want to get a list of `SoftwarePackageIDOverrideTuple` objects.

getSoftwarePackageList

Gets a list of `Software Package` objects. A software package represents an installable chunk of software, such as an RPM.

Result

SOAP Data Type: `SoftwarePackageList`

Description:

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	The ID of the node from which you want to get a list of `SoftwarePackage` objects.

move

Moves a node from one location to another. When you move a node to a new destination node, the moved node becomes a child of the destination node. (The destination node becomes the parent node.) When you move a node, all of that node's dependencies and all servers attached to that node remain intact. All other attributes of the node are inherited from the new parent and all direct attachments remain intact.

Some restrictions on moving a node:

- You can only move nodes within the following Application stacks: Application Servers, DatabaseServer, OS Extras, Other Applications, System Utilities, and WebServer.
- You cannot move a node to itself or to its child.
- You cannot move a locked node to a destination node that is not locked, but you can move an unlocked node to a locked node. If you want to move a locked node to another locked node, you need locking permissions to do so.
- You cannot move a top-level node (such as Application Servers, OS Extras, Web Servers, and so on).
- You can only move a node within its current stack.

Result

SOAP Data Type: `string`

Description: A message describing the results of moving a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node you are moving.
`newParentNodeID`	`decimal`	ID of the parent under which you want to move the node.

setCustomers

Associates a list of customers with a `Node` object. A `Customer` object represents a customer entity in the Opsware System, and consists of such entities as a customer ID, customer name, display name, status, authorization domain, and so on.

The `Customer` object determines who can view and edit the node (with the appropriate customer permissions). After using this operation, the existing list of associated customers will be overwritten.

Result

SOAP Data Type: `string`

Description: A message describing the result of associating the customers with the Node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node you are associating with customers.
`customerIDs`	`decimal[]`	An array of customer IDs to associate with a node.

setPlatforms

Associates a list of platforms (operating systems) with a node. (For a complete list of supported operating systems, please refer to the Opsware System User's Guide.) After using this operation, the existing list of associated platforms will be overwritten.

Result

SOAP Data Type: `string`

Description: A message describing the result of associating the platforms with the Node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node you are associating platforms with.
`platformIDs`	`decimal`	An array of platform IDs to associate with a node.

setSoftwarePackages

Attaches software packages to a node. A software package represents an installable chunk of software, such as an RPM. This operation overwrites the existing list of attached software packages. This operation assumes that the list of packages (`SoftwarePackageIDOverrideTuple[])` is ordered in the sequence required for installation.

Result

SOAP Data Type: string

Description: A message describing the results of attaching software packages to a node.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`nodeID`	`decimal`	ID of the node to which you want to attach a software package.
`softwarePackageIDOverrideTuples []`	`softwarePackageIDOverrideTuples`	An array of `softwarePackageIDOverrideTuples` objects. The array is an ordered list that indicates the installation order of inherited software packages.

Chapter 9: Platform Web Service Operations Reference

I N T H I S C H A P T E R

This chapter describes the following SOAP operations:

- get
- getIDs
- getList

In the Opsware System, the term `platform` is used to refer to an operating system. Each operating system has a unique identifier (ID) assigned by the Opsware System associated with a hard-coded `platform` name assigned by the Opsware System. Users can assign a display name to an operating system in the Opsware Command Center, but the operations discussed in this chapter refer only to the Opsware-System-assigned IDs and `platform` names.

get

Returns the `Platform` object whose ID was specified.

`Platform` IDs are assigned by the Opsware System, and are not visible in the Opsware Command Center.

Result

SOAP Data Type: `Platform`

Description: The specific Operating System, such as RedHat Linux AS 2.1.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`platformID`	`decimal`	The ID of the `platform` whose object you want to see.

getIDs

Returns a list of IDs for `Platform` objects in the database.

To see the IDs assigned to all `platforms,` refer to Appendix A, *Opsware Command Line Interface*, in the Opsware System User's Guide.

Result

SOAP Data Type: `decimal[]`

Description: An array of IDs of `Platform` objects.

Argument

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`platformFilter`	`PlatformFilter`	The object that specifies the search criteria for this operation.

getList

Returns a list of `platform` objects.

To see the names of all `platforms`, refer to Appendix A, *Opsware Command Line Interface*, in the Opsware System User's Guide.

Result

SOAP Data Type: `PlatformList`

Description: An object containing a list of `platform` objects.

Argument

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`platformFilter`	`PlatformFilter`	The object that specifies the search criteria for this operation.

Chapter 10: Security Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

- assignAdministratorRole
- assignCustomerPermissions
- assignFacilityPermissions
- assignUsers
- createSecurityRole
- createUser
- getAssignedSecurityRoleNameList
- getCustomerPermissionList
- getFacilityPermissionList
- getSecurityRoleNameList
- getUserNameList
- isAssignedAdministratorRole
- removeSecurityRole
- removeUser
- unassignAdministratorRole
- unassignUsers
- updatePassword

In the Opsware System, a security role (group) is a set of permisssions that can be assigned to a user, a facility, or a customer. In the Opsware Command Center, a security role is called a group. Since users, facilities, and customers exist separate from the security roles they are granted, security roles and these other objects can be manipulated separately. For example, using the Security Web Service, you can delete a role without deleting a customer without deleting the role assigned to the customer. In addition to a role, a user, facility, or customer, may have additional read or write access permissions that can also be granted using the Security Web Service.

In the Opsware system, you assign permissions to a security role and then assign users to that role. Each user can belong to one or more security roles that control the operations a user can perform and data the user can view. If an Opsware user is assigned to more than one security role, that user has the permissions provided by the union of the multiple security roles.

Except for the `updatePasword` operation, all operations in the Security Web Service must be called by a user with the administrative privilege. The user name and password of this administrative user are required as the first parameters of these operations.

assignAdministratorRole

Assigns a specified Opsware user to the Opsware administrator security role (group).

The Opsware administrator can create Opsware users and assins them to a security role. The Opsware administrator can also promote an Opsware user to the Opsware administrators security role.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.

assignCustomerPermissions

Assigns customer permissions to a security role (group). Customer permissions specify which customers are available to Opsware users in the associated security role. In addition, these permissions specify whether Opsware users have read-only or read/write access to configuration settings and data for the customer.

As an Opsware administrator, you create a security role and then assign access permissions to that security role. After you assign permissions, you assign Opsware users to the security role.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of a security role to which the customers listed will be added.
`customerIDs`	`decimal []`	An array of customer IDs whose permissions are to be changed.
`accessList`	`string []`	An array of strings. Allowed values are: READ, WRITE, NONE.

assignFacilityPermissions

Assigns facility permissions to a security role (group). Facility permissions specify which facilities are available to Opsware users in the associated security role. In addition, these permissions specify whether Opsware users have read-only or read/write access to configuration settings and data for equipment installed at these locations.

As an Opsware administrator, you create a security role and then assign access permissions to that security role. After you assign permissions, you assign Opsware users to the security role.

Result

`SOAP Data Type: string`

Description: The string `OK.`

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of a security role to which the facilities listed will be added.
`facilityIDs`	`decimal []`	An array of facility IDs whose permissions are to be changed.
`accessList`	`string []`	An array of strings. Allowed values are: READ, WRITE, NONE.

assignUsers

Assigns Opsware users to a security role. In the Opsware System, you assign permissions to a security role (group) and then assign Opsware users to that security role. If an Opsware user is assigned to more than one security role, that user can access each of the permissions specified in each security role as well as the permissions provided by the union of the multiple security roles.

Result

`SOAP Data Type: string`

Description:The string `OK.`

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of a security role to which the users will be assigned.
`accessList`	`string []`	An array of user names to be assigned to the security role.

createSecurityRole

Creates an empty security role in the Access & Authentication Directory. After creating a security role (group), the Opsware administrator assigns users to the security role.

In addition to the groups that the Opsware administrator creates, the system also comes with two sets of pre-defined groups: the Basic, Intermediate, and Advanced user groups, and the Code Deployment groups. An Opsware administrator can assign a user to one or more security roles.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of the security role to be created.
`description`	`string`	The description of the security role to be created.

createUser

Creates an Opsware user in the Access & Authentication Directory. As an Opsware administrator, you create a security role (group) and then assign access permissions to that security role. After you assign permissions, you assign Opsware users to that security role. If an Opsware user is assigned to more than one security role, that user has the permissions provided by the union of the multiple security roles.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`userName`	`string`	The login name of the new Opsware user.
`userPassword`	`string`	The login password of the new Opsware user.
`firstName`	`string`	The first name of the new Opsware user.
`lastName`	`string`	The last name of the new Opsware user.
`emailAddress`	`string`	The email address of the new Opsware user.

getAssignedSecurityRoleNameList

Returns a list of names of all the security roles (groups) assigned to an Opsware user.

Result

`SOAP Data Type: string []`

Description: An array of security role names.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`userName`	`string`	The name of an Opsware user with administrative privileges.
`password`	`string`	The password of an Opsware user with administrative privileges.

getCustomerPermissionList

Returns a list of all the customer IDs assigned to the specified security role (group).

Result

`SOAP Data Type: permissionList`

Description: An array of `permission` objects, which specifiy read and write access for customers.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of the security role of the permissions to retrieve.

getFacilityPermissionList

Returns a list of all the facility IDs assigned to the specified security role (group).

Result

`SOAP Data Type: permissionList`

Description: An array of `permission` objects, which specifiy read and write access for customers.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of the security role of the permissions to retrieve.

getSecurityRoleNameList

Returns a list of names of all the security roles (groups) that exist.

Result

`SOAP Data Type: string[]`

Description: An array of security role names.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.

getUserNameList

Returns a list of all Opsware user names that have been assigned to a specified security role (group).

Result

`SOAP Data Type: string []`

Description: An array of Opsware user names.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of the security role of the retrieved users.

isAssignedAdministratorRole

Indicates whether or not the specified Opsware user has been assigned to the Opsware security role.

Result

`SOAP Data Type: boolean`

Description: The result of the test performed by this operation.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`userName`	`string`	The name of the Opsware user that this operation checks for administrator privileges.

removeSecurityRole

Removes a security role (group) along with its permissions.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of the security role to be removed.

removeUser

Removes the specified Opsware user from the Access & Authentication Directory.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`userName`	`string`	The name of the Opsware user to be removed.

unassignAdministratorRole

Unassigns specified Opsware users from the Opsware administrators security role.

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`userNames`	`string []`	An array of Opsware user names to be removed from the administrator role.

unassignUsers

Unassigns users from the specified security role (group).

Result

`SOAP Data Type: string`

Description: The string `OK`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`adminUserName`	`string`	The name of an Opsware user with administrative privileges.
`adminPassword`	`string`	The password of an Opsware user with administrative privileges.
`roleName`	`string`	The name of the security role from which Opsware users are to be unassigned.
`userNames`	`string []`	An array of Opsware user names to be unassigned from a security role.

updatePassword

Changes the password for a given Opsware user.

Result

`SOAP Data Type: string`

Description: The string `true` if the password change was successful.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`userName`	`string`	The name of the Opsware user whose password is to be changed.
`oldPassword`	`string`	The Opsware user's current password.
`newPassword`	`string`	The Opsware user's new password.

Chapter 11: Server Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

- associateWithCustomer
- delete
- get
- getAgentStatus
- getChangeLogList
- getComponents
- getComponentsList
- getEventList
- getIDs
- getList
- getReachableServer
- reconcile
- update

The Server Web Service allows you to perform a vaeriet of operations on an Opsware manager server, such as associating a server with a customer, check the status of the Opsware Agent on the server, get a list of changes for a server, and even reconcile servers.

associateWithCustomer

Associates a server with a customer. A `Customer` object represents a customer entity in the Opsware System, and consists of such entities as a customer ID, customer name, display name, status, authorization domain, and so on.

Result

SOAP Data Type: `string`

Description: A text message that indicates the results of associating a server with a customer (success or failure).

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server to be associated with the customer.
`customerID`	`decimal`	The ID of the customer to be associated with the server.

delete

Deletes the specified server from the Model Repository database (“The Truth”). When a server is deleted from the Model Repository, it is no longer recognized by the Opsware system.

Result

SOAP Data Type: `string`

Description: A text string that indicates the server has been deleted.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server to be deleted.

get

Gets the Server object specified by the server ID.

Result

SOAP Data Type: `Server`

Description: An object that contains about 30 attributes, such as server ID, IP address, agent status, MID, and so on. Each server is identified by the Opsware system by a unique primary ID.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server to get.

getAgentStatus

Gets the status of the agent on the server specified by server's ID. A server is monitored by the Opsware system through the Opsware Agent. If the agent is unreachable then the server is no longer being managed – in other words, the Opsware System cannot access the server. This operation allows you to check the reachability of the agent on the specified server.

Result

SOAP Data Type: `string`

Description: Returns a message stating the current state of the agent: either `OK` or `unreachable`.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server which has the agent you want to get the status of.

getChangeLogList

Returns a list of activity log items for the specified server, during a time window specified by the start and end time-points. The activity log capture changes that a user makes to a server, such as changing its name or its network information, adding or deleting attributes, adding or removing a server from a node, running a Communication Test against the server, running a reconcile on the server, and so on. This information is stored in the change log list (also referred to as “Server History”) and is retrievable using this operation.

Result

SOAP Data Type: `ServerChangeLogList`

Description: A list that includes any changes made to the server during the specified time period.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server from which you want to get a change log list.
`start`	`dateTime`	The start time of the period you want to retrieve for the server change log list.
`end`	`dateTime`	The end time of the period you want to retrieve for the server change log list.

getComponents

Gets the components of a `Server` object specified by ID. Components of a server include the `Customer` object that owns the server, the Facility object in which the server is located, a list of `CpuComponent` objects for all the CPU's on that server, a list of `MemoryComponents`, a list of `StorageComponents`, a list of `InterfaceComponents`, a list of service-level `Node` objects, and so on.

Result

SOAP Data Type: `ServerComponents`

Description: `Server` object "components" from the specified server including CPU, storage, memory, interface, service level, customer, facility, and so on.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server from which you want to get components from the server's Server object.

getComponentsList

Gets a list of `ServerComponents` objects. Components of a server include the `Customer` object that owns the server, the `Facility` object in which the server is located, a list of `CpuComponent` objects for all the CPU's on that server, a list of `MemoryComponents`, a list of `StorageComponents`, a list of `InterfaceComponents`, a list of service-level `Node` objects, and so on.

The maximum number of `ServerComponents` objects that can be retrieved at a time is 200. This limit is enforced to avoid memory issues because of the size of the `ServerComponents` object. To retrieve more than 200 objects, a client application would retrieve the list of server IDs, and iterate to retrieve chunks of 200 objects at a time.

This operation uses the same search criteria and algorithm as the getList operation.

Result

SOAP Data Type: `ServerComponentsList`

Description: Returns a list of ServerComponent objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverFilter`	`ServerFilter`	The object that specifies the search criteria for this operation.

getEventList

Gets a list of server events for `Server` objects that have been added, modified, or removed since a specified time.

The Server Web Service maintains a sliding window (with a default size of two hours) of events describing changes to `Server` objects. This window makes it possible for you to update a client-side cache of `Server` objects without having to retrieve all of the objects. For example, you could maintain a client-side cache as follows:

1. Initially, invoke the `getList` operation to retrieve all relevant `Server` objects.
2. At specified time intervals within the two hour time window, perform these steps:
   - invoke the `getEventList` operation to retrieve the events for the `Server` objects that have been changed.
   - Update the client-side cache with the information returned by `getEventList`.

If you do not update the client-side cache for more than two hours, a full retrieval will be required to obtain all changes. You can set the size of the sliding window for optimal performance by changing the Web Services Data Access Engine configuration file. (For instructions on changing the configuration file, see the *Opsware System 4.7 Administration Guide*.)

There are three types of server events: `ADDED`, `MODIFIED`, or `REMOVED`. If an event is of type `ADDED` or `MODIFIED`, the `ServerEvent` object embeds the Server object. If the event is of type `REMOVED`, the `ServerEvent` object embeds the server ID.

Note: The list of `ServerEvent` objects returned may contain an `ADDED` event, a `MODIFIED` event, as well as a `REMOVED` event for the same server. However, duplicate items for events of the same type are removed. Therefore, the list can contain only one `MODIFIED` event for a server, even if the server has been modified several times within the two hour time window.

Result

SOAP Data Type: `ServerEventList`

Description: A list of server events (additions, modifications, deletions) from the cache (sliding window) maintained by the Server Web Service.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverFilter`	`ServerFilter`	The object that specifies the search criteria for this operation. `ServerFilter.lastRequestTimeStamp` must be specified. Other `ServerFilter` elements are ignored.

getIDs

Returns an array of server IDs that matches the criteria specified by the `serverFilter` argument.

To search for all servers, specify a null `serverFilter`. To search for a single server, specify either the `serverHostname`, `serverIPAddress`, or `machineID` in the `serverFilter`. To search for multiple servers, specify the `serverFilter` elements listed in the last `else` clause of the below algorithm. For example, to search for servers associated with certain customers, specify `serverFilter.customerIDs`.

This operation searches for servers with the following algorithm:

```
if serverFilter is null
return all server IDs
else if serverFilter.serverIDs is not empty
throw an exception
else if server.serverHostname is not null
return server ID by name
else if serverFilter.serverIPAddress is not null
return server ID by IP
else if serverFilter.machineID is not null
return server ID by MAC
else
return server IDs by the conjunction (logical AND) of the
following serverFilter elements:
customerIDs
facilityIDs
agentStatusFilters
deploymentStageFilters
serverUseFilters
```

Result

SOAP Data Type: `decimal[]`

Description: An array of server IDs.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverFilter`	`ServerFilter`	The object that specifies the search criteria for this operation.

getList

Returns a list of `Server` objects that matches the criteria specified by the `serverFilter` argument.

To search for all servers, specify a null `serverFilter`. To search for a single server, specify either the `serverHostname`, `serverIPAddress`, or `machineID` in the `serverFilter`. To search for multiple servers, specify the `serverFilter` elements listed in the last `else` clause of the below algorithm. For example, to search for servers associated with certain customers, specify `serverFilter.customerIDs`.

This operation searches for servers with the following algorithm:

```
if serverFilter is null
throw an exception
else if serverFilter.serverIDs is not empty
return servers by IDs
else if server.serverHostname is not null
return server by name
else if serverFilter.serverIPAddress is not null
return server by IP
else if serverFilter.machineID is not null
return server by MID
else
return the servers that match the conjunction (logical AND)
of the following serverFilter elements:
customerIDs
facilityIDs
agentStatusFilters
deploymentStageFilters
serverUseFilters
```

Result

SOAP Data Type: `ServerList`

Description: An array of `Server` objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverFilter`	`ServerFilter`	The object that specifies the search criteria for this operation.

getReachableServer

Gets the reachable Server object that matches the specified IP address. A server is reachable if the Opsware core can communicate with the Opsware Agent installed on the server.

Result

SOAP Data Type: `Server`

Description: An object that contains about 30 attributes, such as server ID, IP address, agent status, MID, and so on. If the server is reachable, this operation will get the specified `Server` object.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`ipAddress`	`string`	IP address of the server.

reconcile

Starts a reconcile session and returns the reconcile session ID. When you reconcile a server, you are updating the actual software configuration of a server based on the specified configuration stored in the Model Repository ("The Truth").

Result

SOAP Data Type: `decimal`

Description: The reconcile session ID (if successful).

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`ReconcileSession Argument`	`reconcileSessionArgument`	<need to add>

update

Updates the attributes of the specified server. A server's attributes are used to describe:

- How the server is being used. By default: not specified, production, and staging.
- What stage of the life cycle the server is in (decommissioned, in deployment, or live.
- How the end user wants to describe the server.

Result

SOAP Data Type: `string`

Description: A list of all servers that have had their attributes updated.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server whose attributes you want to update.
`serverUse`		Server use flag of server to update.
`deploymentStage`		Deployment stage flag of server to update
`serverDescription`		Description of server to update. (A short name for the description.)
`notes`		Comments of the server to update.

Chapter 12: Software Web Service Operations Reference

IN THIS CHAPTER

This chapter describes the following SOAP operations:

- get
- getAssociatedList
- getIDs
- getInstalledList
- getList

The Software Web Service operates on the Opsware entities that represent software packages.

In the Opsware System, software packages reside in a central Software Repository. The software function is used to install and uninstall software packages on servers. You can select software packages from the Software Repository, select the servers that you want to install the software on, preview the results of the installation, and install the software all in a single operation.

get

Returns a software package object specified by ID.

Result

SOAP Data Type: `SoftwarePackage`

Description: The object that represents the software package.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`softwarePackageID`	`decimal`	The ID (primary key) of the software package to be returned.

getAssociatedList

Returns software packages associated with a server.

Result

SOAP Data Type: `SoftwarePackageList`

Description: An array of software package objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The ID of the server associated with a software package.

getIDs

Returns an array of software package IDs that match the criteria specified by the `softwarePackageFilter` argument. For example, to search for software packages associated with one or more platforms (operating sytems), for example, specify `softwarePackageFilter.platformIDs`. To search by both platforms and customers (logical AND), specify the `platformIDs` and `customerIDs` elements of the `softwarePackageFilter`. The `softwarePackageFilter.softwarePackageIDs` array must be null or empty.

```
if softwarePackageFilter is null
throw an exception
else if softwarePackageFilter.softwarePackageIDs is non-empty
throw an exception
else
return an array of software package objects matching the
conjunction of the following SoftwarePackageFilter elements:
customerIDs, platformIDs, softwarePackageName. Null and empty
elements are ignored.
```

Result

SOAP Data Type: `Decimal[]`

Description: An array of software package IDs.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`softwarePackageFilter`	`SoftwarePackageFilter`	The object that specifies the search criteria for this operation.

getInstalledList

Returns software packages installed on a server.

Result

SOAP Data Type: `InstalledSoftwarePackageList`

Description: An array of software objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`Decimal`	The ID of the server on which the software packages are installed.

getList

Returns an array of software package objects that match the criteria specified by the `softwarePackageFilter` argument. To search for software packages by their IDs (primary keys), specify `softwarePackageFilter.softwarePackageIDs`. To search by the other `softwarePackageFilter` elements, specify a null or empty `softwarePackageFilter.softwarePackageIDs`. To search for software packages associated with a particular customer, for example, specify `softwarePackageFilter.customerIDs`.

This operation searches for software packages with the following algorithm:

```
if softwarePackageFilter is null
throw an exception
else if softwarePackageFilter.softwarePackageIDs is non-empty
return corresponding array of software package objects.
else
return an array of software package objects matching the
conjunction of the following SoftwarePackageFilter elements:
customerIDs, platformIDs, softwarePackageName. Null and empty
elements are ignored.
```

Result

SOAP Data Type: `SoftwarePackageList`

Description: An array of software package objects.

Arguments

ARGUMENT NAME	SOAP DATA TYPE	DESCRIPTION
`SoftwarePackageList`	`SoftwarePackageFilter`	The object that specifies the search criteria for this operation.

Chapter 13: Complex Data Types Reference

I N   T H I S   C H A P T E R

This chapter describes the complex SOAP data types used by the Opsware System Web Services API 2.0.

CommandSession

Provides configuration information about Command Engine sessions.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`agentName`	`string`	Not used; to be removed in a future release.
`endDate`	`dateTime`	Date and time when the Command Engine session ended.
`hostName`	`string`	Hostname and realm of the server from where the Command Engine session was initiated.
`miscDigest`	`string`	Not used; to be removed in a future release.
`myJobsVisible`	`boolean`	Indicates if the job is displayed in the My Jobs panel of the Opware Command Center.
`ownerName`	`string`	Not used; to be removed in a future release.
`paramsDigest`	`string`	Not used; to be removed in a future release.
`resultsDigest`	`string`	Not used; to be removed in a future release.
`scheduleDate`	`dateTime`	Time and date the Command Engine session is scheduled to be run.
`scheduleId`	`decimal`	Not used; to be removed in a future release.
`scriptVersionId`	`decimal`	Version number of the Command Engine session script.
`serviceInstanceId`	`decimal`	Command Engine instance which ran (or will run) the current session.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`sessionDescription`	`string`	The text describing the Job Name displayed in the My Jobs panel of the Opsware Command Center.
`sessionId`	`decimal`	Unique identifier for each Command Engine session.
`signature`	`string`	Signature verification of the session and all auxiliary data.
`signOffDate`	`dateTime`	Not used; to be removed in a future release.
`staleDate`	`dateTime`	The point in time at which a Command Engine session will expire if the session has not been started or completed.
`startDate`	`dateTime`	Date and time when the Command Engine session started.
`status`	`string`	Status of the Command Engine session. Examples: `SUCCESS`, `FAILED`.
`userId`	`decimal`	Unique identifier of the Opsware user who initiated the Command Engine session.
`userName`	`string`	Not used; to be removed in a future release; replaced by `userId`.

CustomAttribute

Represents a name-value pair that can be assigned to an Opsware entiy such as a customer, server, software application (node), or facility.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`key`	`string`	The name of the custom attribute.
`objectID`	`decimal`	The unique ID of the Opsware entity the custom attribute has been assigned to. For example, if the `objectType` is `CUSTOMER`, then the `objectID` is the same as the `customerID`.

`objectType`	`string`	The type of Opsware entity the custom attribute has been assigned to. The `objectType` may be one of the following:
		`CUSTOMER`
		`FACILITY`
		`NODE`
		`SERVER`

`value`	`string`	The value of the custom attribute.

CustomAttributeFilter

Determines the search results of the Custom Attribute Service `getValues`, `remove`, and `set` operations.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customAttributeKeys`	`string[]`	An array of custom attribute names.
`objectIDs`	`decimal[]`	The unique ID of the Opsware entity the custom attribute has been assigned to. For example, if the `objectType` is `CUSTOMER`, then the `objectID` is the same as the `customerID`.

`objectType`	`string`	The type of Opsware entity the custom attribute has been assigned to. The `objectType` may be one of the following:
		`CUSTOMER`
		`FACILITY`
		`NODE`
		`SERVER`

Customer

Represents a customer entity in the Opsware System.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`authorizationDomain`	`string`	The authorization domain of the customer, usually the same as the domain name.
`businessAccountID`	`decimal`	TBD
`customerID`	`decimal`	The unique ID (primary key) that identifies the customer.
`customerName`	`string`	The short name of the customer.
`displayName`	`string`	The display name of the customer in the Opsware Command Center.
`status`	`string`	The status of the customer. Allowed values: `ACTIVE` `(TBD others?)`

CustomerFilter

Determines the search results of the Customer Web Service `getIDs` and `getList` operations.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`facilityID`	`decimal`	ID of the facility associated with customers to retrieve.
`customerName`	`string`	The name of the customer to retrieve.
`customerIDs`	`decimal[]`	An array of customer IDs to retrieve.

DistributedScriptArgument

The argument of the `execute` operation of theDistributed Script Web Service, which enables you to run and monitor a script on multiple servers.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerIDs`	`decimal[]`	An array of customer IDs associated with the servers on which the distributed script will be executed.
`hiddenParameters`	`DistributedScriptParameterSet`	Includes the encrypted parameters such as the password.
`myJobsVisible`	`boolean`	Indicates if the distributed script execution job is visible in the My Jobs panel of the Opsware Command Center. Returns true if the job should be visible.
`parameters`	`DistributedScriptParameterSet`	The parameters required for the distributed script to be executed. For Unix scripts, use Bourne (`sh`) shell syntax and for Windows scripts, use `cmd.exe` parameter syntax.
`scheduleDate`	`dateTime`	The date on which to run the script. If the schedule date is not specified, the script is executed immediately.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`scriptVersion`	`string`	The version information of the distributed script to be executed. Each time a script is modified, new script version information is created and stored.
`sessionDescription`	`string`	The text describing the Job Name in the My Jobs pannel in the Opsware Command Center.
`staleDate`	`dateTime`	The time after which it is too late to run a scheduled distributed script. If the `staleDate` is later than the current date, the script will not be run.
`userName`	`string`	The name of the Opsware user who executed the distributed script.

DistributedScriptParameterSet

Contains the details for the parameter of a distributed script execution (DSE) operation on a managed server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`scriptArguments`	`string`	The arguments of the distributed script to be executed.
`scriptCodeType`	`string`	The type of distributed script to be executed. The types include Unix shell, Windows.BAT, or Windows VBScript.
`scriptID`	`decimal`	The ID of the distributed script to be executed.
`scriptName`	`string`	The name of the distributed script be to executed.
`scriptText`	`string`	The contents of the distributed script to be executed.
`serverDomain`	`string`	The Windows domain of the server in which the script will be executed.
`serverIDs`	`decimal[]`	An array unique IDs of the servers on which the distributed script will be executed.
`serverPassword`	`string`	The password for the server on which the distributed script will be executed.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverUserName`	`string`	The user name for the server on which the distributed script will be executed.
`stderrByteSize`	`decimal`	The number of bytes in the `stderr` generated by the script.
`stdoutByteSize`	`decimal`	The number of bytes in the `stdout` generated by the script.
`waitTime`	`int`	The wait time in minutes before timing out the distributed script execution operation. This value is the amount of time a script has to complete execution on a server. If the script is not finished when the time-out value is reached, the script is stopped by the Opsware System and a script error occurs.

DSESessionProgress

Represents the progress or current status of a distributed script execution (DSE) session on managed servers.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`doneServers`	`decimal`	The number of servers that have completed the distributed script execution operation.
`doneSteps`	`decimal`	The number of completed steps for the distributed script execution operation.
`errorServers`	`DSEServerStatus[]`	The servers on which the error occured during distributed script execution operation.
`hostprogress`	`DSEServerProgress[]`	An array of elements, each of which represents the distributed script execution progress of a specific server.
`totalServers`	`decimal`	The total number of servers on which the distributed script execution operation is running.
`totalSteps`	`decimal`	The total number of steps required for the distributed script execution operation.

DSESessionResult

Represents the results of a distributed script execution (DSE) session on managed servers.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`errorServers`	`DSEServerStatus[]`	The servers on which the error occured during distributed script execution operation.
`hostsResults`	`DSEServerResult[]`	An array of elements, each of which represents the distributed script execution results of a specific server.
`totalServers`	`decimal`	The total number of servers on which the distributed script execution operation is running.

DSEServerProgress

Represents the progress or current status of a distributed script execution (DSE) session on a specific server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`doneSteps`	`decimal`	The number of completed steps for the distributed script execution operation.
`errorDetail`	`string`	The details of an error that might occur during distributed script execution operation.
`errorName`	`string`	The name of an error that might occur during distributed script execution operation.
`serverID`	`decimal`	The unique identifier (primary key) of the server on which the script will be executed.

`stage`	`string`	The deployment stage of the server. By default, Opsware is installed with these possible values for the deployment stage:
		`IN DEPLOYMENT`
		`LIVE`
		`NOT SPECIFIED`
		`OFFLINE`
		`OPS READY`

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`status`	`string`	The status of the agent on the server specified by the server ID. Allowed values: `OK` - The server is managed by the Opsware System `NOT REACHABLE` - The Server is unmanageable by the Opsware System due to an error.
`totalSteps`	`decimal`	The total number of steps required for the distributed script execution operation.

DSEServerResult

Represents the results of a distributed script execution (DSE) session on a specific server.

`DSEServerResult` extends `DSEServerStatus` and has no sub elements other than those inherited from `DSEServerStatus`.

DSEServerStatus

Represents the status of the distributed script execution (DSE) on a specific server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`commandID`	`decimal`	The identifier associated with the task running the distributed script.
`errorDetail`	`string`	The details of an error that might occur during script execution operation.
`errorName`	`string`	The name of an error that might occur during script execution operation.
`serverID`	`decimal`	The unique identifier (primary key) of the server on which the script will be executed.
`status`	`string`	The status of the agent on the server specified by the server ID. Allowed values: `OK` - The server is managed by the Opsware System `NOT REACHABLE` - The Server is unmanageable by the Opsware System due to an error.

EventBase

Specifies the type of event that has occurred. This type is extended by `ServerEvent`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION

`eventType`	`string`	Specifies what action has occurred. Allowed values for a `ServerEvent`:
		`ADDED`
		`MODIFIED`
		`REMOVED`

Facility

Represents a collection of servers stored in a single Opsware Model Repository (database). A facility usually corresponds to a specific data center or geographical location (city) of an Opsware System installation.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`displayName`	`string`	The name of the facility displayed by the Opsware Command Center.
`facilityID`	`decimal`	The unique identifier (primary key) of the facility.
`facilityName`	`string`	The short name of the facility.
`status`	`string`	The facility's status. Allowed values: `ACTIVE` `DECOMMISSIONED`.

FacilityFilter

Determines the search results of the Facility Web Service `getIDs` and `getList` operations.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerID`	`decimal`	ID of the customer associated with the facilities to retrieve.
`facilityIDs`	`decimal[]`	An array of facility IDs to retrieve.

InstalledSoftwarePackage

Represents a software package that has been installed on a managed server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The unique identifier (primary key) of the server on which the software package is installed.
`softwarePackageDescription`	`string`	An informational field describing the package.
`softwarePackageDisplayName`	`string`	The name of the software package displayed in the Opsware Command Center.
`softwarePackageID`	`decimal`	The unique identifier (primary key) for the `SoftwarePackage` object.
`softwarePackageName`	`string`	The name of the software packaged used internally by the Opsware System.
`softwarePackageType`	`string`	The type of the package, such as HOTFIX, RPM, MSI, LPP, APAR, ZIP.
`softwareRelease`	`string`	For RPMs only, the release info generated by: `rpm -q.`
`softwareVersion`	`string`	For RPM, the version info generated by: `rpm -q`. For MSI, the version specified by the user.

ListBase

The base type for many of the `<type>List` complex types. For example, `CustomerList` and `FacilityList` extend `ListBase`. Each `<type>List` complex data type embeds an array of particular complex objects. For example, the `CustomerList` embeds an array of `Customer` objects.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`timeStamp`	`dateTime`	Usually the date and time the list was retrieved.

Node

Represents an element in the Opsware System software tree.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`allowDevice`	`boolean`	Indicates whether a server can be attached to the node.
`allowMultiple`	`boolean`	Indicates whether the stack allows a device to be attached more than once. For example, a server can only be attached to a node in the operating system stack once.
`description`	`string`	A text description of the node displayed in the Opsware Command Center.
`fullName`	`string`	The concatenation of all short names down the path to this node.
`location`	`string`	A modified version of the node's synthetic name, which specifies the full path to this node. For some stacks, prefix elements such as `AppServer` may be omitted, for example, `WebLogic/SunOS/5.7/8.0`
`nodeID`	`decimal`	The unique identifier (primary key) of the node.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`notes`	`string`	Notes displayed in the Opsware Command Center.
`parentNodeID`	`decimal`	The identifier of the node's parent in the tree.
`reconcileOrder`	`decimal`	The default reconciliation (installation) order of the package attached to the node. This default can be overridden with install order dependencies.
`shortName`	`string`	An internal node name used by the Opsware System. The value must be all upper case and contain no spaces.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`stackID`	`decimal`	The unique ID of a stack (subtree) in the node tree. For example, the ID of the System Utilities stack is 7. Valid IDs are: 1 - customer 2- facility 3 - hardware 4 - service levels 5 - operating system 6 - opsware 7 - System Utilities 8 - DatabaseServer 9 - AppServer 10 - WebServer 11 - Custom Apps 12 - OS Extras 13 - non-existent 14 - templates 15 - patches
`status`	`string`	Has vlaues such as `ACTIVE` or `DEPRECATED`. The status is not used for software nodes.
`syntheticName`	`string`	The path to the node (except for the top level) displayed by the Opsware Command Center.

NodeFilter

Determines the search results of the Node Web Service `getIDs` and `getList` operations. A node is an element in Opsware's software tree.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerIDs`	`decimal[]`	An array of customer IDs associated with the nodes to retrieve.
`nodeIDs`	`decimal[]`	An array of node IDs to retrieve.
`nodeName`	`string`	The name of the node to retrieve.
`platformIDs`	`decimal[]`	An array of platform IDs. A platform is the name and version of an operating system.
`stackIDs`	`decimal[]`	The unique ID of a stack (subtree) in the software model tree. For a list of valid IDs, see the `stackID` description of the Node data type.

Permission

Represents the security permission (read or write access) of an Opsware customer or facility.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`access`	`int`	The type of access. Allowed values: `0` (none) `1` (read) `2` (read-write)
`ID`	`string`	The identifier (primary key) of the customer or facility.

Platform

Represents the definition of an operating system in the Opsware System. A platform is the name and version of an operating system (for example, Red Hat Linux 4.2).

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`architecture`	`string`	The type of hardware, such as X86 or SPARC.
`isPatchable`	`boolean`	A flag is used in the Install Patch Wizard of the Opsware Command Center. If `isPatchable` is false, then the platform is not among the platforms the the user can select.
`osVersion`	`string`	Not used; to be removed in a future release.
`platformDescription`	`string`	Descriptive text or notes for the platform.
`platformID`	`decimal`	The unique identifier (primary key) of the platform.
`platformName`	`string`	The name displayed in the Opsware Command Center. For example: Red Hat Enterprise Linux AS 2.1
`platformShortName`	`string`	An internal name used by the Opsware System. The value must be all upper case and contain no spaces.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`platformVersion`	`string`	An internal name used by the Opsware System to match what the agents report with a known platform object.

PlatformFilter

Determines the search results of the Platform Web Service `getIDs` and `getList` operations. A platform is the name and version of an operating system (for example, Red Hat Linux 4.2).

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`platformIDs`	`decimal[]`	An array of platform IDs to retrieve.
`platformName`	`string`	The name of the platform to retrieve.

ReconcilePackageBase

Contains the name of a the software package affected by the reconcile operation. This type is extended by `ReconcilePackageDetails`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`displayName`	`string`	The name of the software package displayed in the Opsware Command Center.
`uniqueName`	`string`	The name of the software packaged used internally by the Opsware System.

ReconcilePackageDetails

Contains information about the software package that is installed or uninstalled by a reconcile operation. `ReconcilePackageDetails` extends `ReconcilePackageBase` and is extended by `ReconcilePackageOutput`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`installFlags`	`string`	Optional arguments passed to the installer on the managed server.
`installOrder`	`decimal`	The default reconcile (installation) order of the package attached to the node. This default can be overridden with install order dependencies.
`messages`	`string[]`	Any informational messages associated with a package that the reconcile may generate.
`nodeIDs`	`decimal[]`	The identifiers of the nodes to which the software package has been attached.
`packageID`	`decimal`	The unique identifier (primary key) of the `SoftwarePackage` object.
`packageName`	`string`	The name of the package being installed or uninsalled.
`packageRelease`	`string`	For RPMs only, the release info generated by: `rpm -q`

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`packageType`	`string`	The type of the package, such as HOTFIX, RPM, MSI, LPP, APAR, ZIP.
`packageVersion`	`string`	For RPMs, the version info generated by: `rpm -q`. For MSIs, the version specified by the user.
`rebootOnInstall`	`boolean`	Indicates whether to reboot the managed server after successful installation of the software package.
`rebootOnUninstall`	`boolean`	Indicates whether to reboot the managed server after successful uninstallation of the software package.
`removeFlags`	`string`	Optional arguments passed to the uninstaller on the managed server.
`uninstallable`	`boolean`	Specifies whether the Opsware System has determined that the package should not be installed.

ReconcilePackageOutput

The output returned by a reconcile operation. This type extends `ReconcilePackageDetails`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`errorDetail`	`string`	Details of any errors generated by the reconcile.
`errorName`	`string`	The name of an error generated by the reconcile.
`installedDate`	`dateTime`	The date and time when the package was installed by the reconcile.
`output`	`string`	The output generated by the tool that performed the reconcile. The tool varies with operating system. Examples of these tools are: rpm, msi, patchrm, pkgadd.
`result`	`integer`	The return code from the install or remove operation.

ReconcileServerParameterSet

Contains the details for the parameter of a reconcile operation on a managed server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`debug`	`boolean`	Specifies whether to run the reconcile in debug mode.
`installAll`	`boolean`	Specifies whether to install all software packages attached to the nodes to which the server has been attached.
`installNodeIDs`	`decimal[]`	The identifiers of the nodes to which the software packages and server have been attached.
`removeAll`	`boolean`	Specifies whether to uninstall all software packages attached to the application nodes from which the server has been dettached.
`removeNodeIDs`	`decimal[]`	The identifiers of the application nodes from which software packages have been detached.
`serverID`	`decimal`	The unique identifier (primary key) of the server on which the reconcile operation will be applied.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`templateID`	`decimal`	The identifier of the Opsware template that contains the software to be installed or uninstalled.
`testMode`	`boolean`	Specifies whether this reconcile operation is a preview (test). In a preview reconcile , no software is actually installed or uninstalled.

ReconcileServerProgress

Represents the progress or current status of a reconcile session on a specific server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`completedTasks`	`decimal`	The number of completed steps for this reconcile operation. Steps include: downloading a package to the managed server, installing a package, and re-registering the installed package with the Opsware System.
`message`	`string`	A short text description of the current state of the reconcile operation. Example: “Querying Opsware Agent for installed packages.”
`serverID`	`decimal`	The unique identifier of the managed server on which the reconcile operation is running.
`totalTasks`	`decimal`	The total number of steps required for the reconcile operation.

ReconcileServerResult

Indicates which software packages were installed or removed by the reconcile operation on a specific server. This type extends `ReconcileServerResultBase`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`installedAsSideEffect`	`ReconcilePackageBase[]`	An array that represents the software packages installed as a side effect on the server by the reconcile operation. Examples of side effects are APARs installed when an UpdateFileset is installed, or a Solaris patch included in a Solaris package.
`removedAdSideEffect`	`ReconcilePackageBase[]`	An array that represents the software packages removed as a side effect of the reconcile operation.
`wasInstalled`	`ReconcilePackageOutput[]`	An array that represents the software packages successfully installed on the server by the reconcile operation.
`wasNotInstalled`	`ReconcilePackageOutput[]`	An array that represents the software packages that should have been installed on the server by the reconcile operation but were not.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`wasNotRemoved`	`ReconcilePackageOutput []`	An array that represents the software packages that should have been removed from the server by the reconcile operation but were not.
`wasRemoved`	`ReconcilePackageOutput []`	An array that represents the software packages successfully removed from the server by the reconcile operation.

ReconcileServerResultBase

Contains information about the results of a a reconcile operation on a specific server. This type is extended by `ReconcileServerResult` and `ReconcileServerResultPreview`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`cannotBeRemoved`	`ReconcilePackageBase[]`	An array of elements that represent the software packages that could not be removed from the server by the reconcile operation. For example, a Fileset might not be removed because all related filesets must be removed at the same time.
`debugError`	`string`	An error message generated by the reconcile operation. Appears only if `ReconcileServerParameterSet.debug` is set to true.
`debugWarning`	`string`	A warning message generated by the reconcile operation. Appears only if `ReconcileServerParameterSet.debug` is set to true
`error`	`string`	An error message generated by the reconcile operation.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverID`	`decimal`	The unique identifier (primary key) of the server that was reconciled.
`status`	`string`	The status of the reconcile operation. Examples: `SUCCESS`, `FAILURE`, `ABORTED`, `BOOTING`, `CYCLING`.
`warning`	`string`	A warning message generated by the reconcile operation.
`willNotInstall`	`ReconcilePackageDetails[]`	An array of elements that represent the software packages that the Opsware System determined should not be installed by the reconcile operation. For example, the package may already be installed, or an UpdateFileset will not be installed because a required BaseFileset does not exist.

ReconcileServerResultPreview

Indicates which software packages the reconcile preview (test) operation has determined would be installed or removed on a particular server. This type extends `ReconcileServerResultBase`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`willBeRemovedAsSideEffect`	`ReconcilePackageBase[]`	An array that represents the software packages that would be removed from the server as a side effect of the reconcile operation.
`willInstall`	`ReconcilePackageDetails[]`	An array that represents the software packages that would be installed on the server by the reconcile operation.
`willRemove`	`ReconcilePackageDetails[]`	An array that represents the software packages that would be removed from the server by the reconcile operation.

ReconcileSessionArgument

The argument of the `reconcile` operation of the Server Web Service, which installs or uninstalls software (applications or patches) on servers managed by the Opsware System.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerIDs`	`decimal[]`	The identifiers of the customers associated with the servers on which the reconcile operation will be applied.
`globalArguments`	`ReconcileServerParameterSet`	The arguments common to all servers on which the reconcile operation will be applied.
`localArgumentsList`	`ReconcileServerParameterSet[]`	The arguments for a single server on which the reconcile operation will be applied. A non-null element of `localArgumentsList` overrides the corresponding element in `globalArguments`.
`serverIDs`	`decimal[]`	The identifiers of the servers on which the reconcile operation will be applied.
`userName`	`string`	The name of the Opsware user performing the reconciliation operation.

ReconcileSessionProgress

Represents the progress or current status of a reconcile session on managed servers.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`completedServers`	`decimal`	The number of servers that have completed the reconcile operation.
`error`	`string`	The message of a session error.
`serversProgress`	`ReconcileServerProgress[]`	An array of elements, each of which represent the reconcile progress of a specific server.
`totalServers`	`decimal`	The total number of servers on which the reconcile operation is running.

ReconcileSessionResult

Represents the results of a reconcile operation on managed servers. This type extends extends `ReconcileSessionResultBase`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serversResult`	`ReconcileServerResult[]`	An array of elements, each of which represents the results of a reconcile operation on a specific server.

ReconcileSessionResultBase

Contains information about a reconcile operation on managed servers. This type is extended by `ReconcileSessionResult` and `ReconcileSessionResultPreview`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`completedServers`	`decimal`	The number of servers on which the reconcile operation has finished.
`error`	`string`	The message of an error resulting from the reconcile session.
`totalServers`	`decimal`	The total number of servers on which the reconcile operation is applied.

ReconcileSessionResultPreview

Represents the results of a preview (test) reconcile operation on managed servers.This type extends extends `ReconcileSessionResultBase`.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serversResultPreview`	`ReconcileServerResultPreview[]`	An array of elements, each of which represents the results of a preview (test) reconcile operation on a specific server

Server

Represents a server machine managed by the Opsware System.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION

`agentStatus`	`string`	Tthe status of the agent on the servers. Allowed values are:
		`OK`
		`NOT REACHABLE`

`assetTag`	`string`	The inventory asset number.
`chassisID`	`string`	A unique hardware-based identifier that the Opsware Agent discovers, typically derived from some property of the server's chassis. For this ID, the Opsware System uses an interface's MAC address, or the hostid on Solaris servers, or the serial number for one of the interfaces.
`chassisSerialNumber`	`string`	The serial number of the hardware chassis.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION

`deploymentStage`	`string`	The server's deployment stage. By default, Opsware is installed with these possible values for the deployment stage:
		`IN DEPLOYMENT`
		`LIVE`
		`NOT SPECIFIED`
		`OFFLINE`
		`OPS READY`

`discoverDate`	`dateTime`	The date and time the server first appeared in the Opsware System.
`effectiveBeginDate`	`dateTime`	The date and time of the most recent hardware registration, which is performed by the Opsware agent.
`loopbackIP`	`string`	The IP address for network loopback testing.
`machineID`	`string`	The MID, a unique ID of the server assigned by Opsware when the server first registers. Stored on the hard disk of the server, the MID is usually the same as the `serverID`.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`managementIP`	`string`	The IP address that the Opsware System uses to communicate with the Opsware Agent on the server. With static NAT, the management IP address for a server will not be the same as the primary IP address. Also with static NAT, the management IP is the NAT-translated IP address for the server. When static NAT is not being used, the management IP address is always the same as the primary IP address.
`notes`	`string`	Descriptive text for the server.

`opswLifecycle`	`string`	Specifies the life cycle phase for the server. Allowed values for a server in the server pool:
		`AVAILABLE`
		`INSTALLING OS`
		`BUILD FAILED`
		Allowed values for managed servers:
		`MANAGED`
		`DEACTIVATED`

`origin`	`string`	Indicates how the server got into the Opsware System. Allowed values: PROVISIONED ASSIMILATED

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`osVersion`	`string`	The version of the operating system definition specified by the Opsware administrator.
`primaryIP`	`string`	The IP address of the management interface. When you change the management interface, the primary IP changes to the IP address of that interface. The primary IP address is a locally-configured IP address.

`reporting`	`string`	The status of the agent's report of the server to the Opsware System. Allowed values:
		`OK`
		`REGISTRATION IN PROGRESS`
		`REPORTING ERROR`

`serialNumber`	`string`	The manufacturer's serial number of the server. If possibl, e Opsware assigns the chassis ID to the serial number.
`serverDescription`	`string`	By default, the same value as `systemName`.
`serverID`	`decimal`	A unique ID (primary key) of the managed server.
`serverMfg`	`string`	The name of the vendor that manufacured the server.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`serverModel`	`string`	The model name of the server.
`serverType`	`string`	Allowed value: `SERVER`

`serverUse`	`string`	By default, Opsware is installed with these possible values for the `serverUse`:
		`STAGING`
		`PRODUCTION`
		`NOT SPECIFIED`
		The CDR subsystem relies on `STAGING` and `PRODUCTION`.

`systemDescriptor`	`string`	Not used for servers.
`systemLocation`	`string`	Specifies the physical location of the server. For example: the facility, floor, and rack number.
`systemName`	`string`	The server name, usually the host name.
`systemObjectID`	`string`	Not used for servers.

ServerChangeLog

Represents an entry in the change history log of a managed server. The log entries include events such as adding a server to a node, installing a template on a server, and the success or failure of a reconciliation (software installation) on a server. For a full list of the events logged, see the section “Server Histories and Reports Overview” in the User’s Guide.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`changeSummary`	`string`	A text description of the operation performed on the server.
`modifiedBy`	`string`	The name of the Opsware user who changed the server.
`modifiedDate`	`dateTime`	The date and time the change was made to the server.
`serverChangeLogID`	`decimal`	The unique ID (primary key) of the `ServerChangeLog` object.
`serverID`	`decimal`	The uniqie ID (primary key) of the changed server.

ServerComponents

Contains a variety of attributes (primarlly hardware) of a managed server.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION

`cpuComponent`	`CpuComponent[]`	Includes these subelements:
		`cacheSize`
		`cpuFamily`
		`cpuModel`
		`cpuSlot`
		`cpuSpeed`
		`cpuVendor`

`customer`	`Customer`	The customer associated with the server.
`facility`	`Facility`	The facility associated with the server.

`interfaceComponents`	`InterfaceComponent[]`	Includes these subelements:
		`cardIndex`
		`cardSerialNumber`
		`cpuSpeed`
		`dramBb`
		`hardwareAddress`
		`interfaceAlias`
		`interfaceDescription`
		`interfaceSpeed`
		`interfaceType`
		`ipAddress`
		`memoryQuantity`
		`netMask`
		`primaryInterface`
		`slot`

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`memoryComponents`	`MemoryComponent[]`	Includes these subelements: `memoryLocation` `memoryQuantity` `memoryType`
`serviceLevelNodes`	`Node[]`	An array of nodes that represent user-defined categories such as Silver, Gold, and Platinum services.
`storageComponents`	`StorageComponent[]`	Includes these subelements: `storageCapacity` `storageDrive` `storageMedia` `storageMfg` `storageModel` `storageType`

ServerEvent

Represents an action that has been performed on a `Server` object. The type of action is specified by the `EventBase`, which `ServerEvent` extends.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`server`	`Server`	The managed server on which the event took place.
`serverID`	`decimal`	The unique ID (primary key) of the `Server` object.

ServerFilter

Determines the search results of the Server Web Service `getIDs`, `getList`, and `getEventList` operations.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION

`agentStatusFilters`	`string[]`	An array of strings representing the status of the agents on the servers to retrieve. Allowed values are:
		`OK`
		`NOT REACHABLE`

`customerIDs`	`decimal[]`	An array of customer IDs associated with the servers to retrieve.

`deploymentStageFilters`	`string[]`	An array of strings representing the deployment stage of the servers to retrieve. By default, Opsware is installed with these possible values for the deployment stage:
		`IN DEPLOYMENT`
		`LIVE`
		`NOT SPECIFIED`
		`OFFLINE`
		`OPS READY`

`facilityIDs`	`decimal[]`	An array of facility IDs associated with the servers to retrieve.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`lastRequestTimeStamp`	`dateTime`	The start date and time of the events returned by `getEventList`. The end date and time is the current date and time. Usually, `lastRequestTimeStamp` is the date and time of the most recent call to `getEventList`.
`machineID`	`string`	The MID, a unique ID for the server. (For details see the machineID description of the Server type.)
`serverHostName`	`string`	The IP host name of the server to retrieve.
`serverIDs`	`decimal[]`	An array of server IDs to retrieve.
`serverIPAddress`	`string`	The IP address of the server to retrieve.

`serverUseFilters`	`string`	The server “use” of the servers to retrieve. By default, Opsware is installed with these values for the server “use”:
		`STAGING`
		`PRODUCTION`
		`NOT SPECIFIED`
		The CDR subsystem relies on `STAGING` and `PRODUCTION`.

SoftwarePackage

Represents an installable piece of software, such as an RPM. Software packages are uploaded into the Opsware Software Repository.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`createDate`	`dateTime`	The date and time the software package was first uploaded into the Opsware Software Repository.
`createdBy`	`string`	The Opsware user who who first uploaded the software package.
`existsOnWord`	`boolean`	Indicates whether or not the software package has been uploaded into the Opsware Software Repository.
`fileSize`	`decimal`	The size of the software package file, in bytes.
`fileType`	`string`	The type of file, such as MSI, RPM, ZIP, EXE, TAR.
`installFlags`	`string`	Optional arguments passed to the installer on the managed server.
`modifiedBy`	`string`	The name of the Opsware user who most recently modified the package properties or who re-uploaded the package.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`modifiedDate`	`dateTime`	The date and time any property of the package is modified or when the package is re-uploaded.
`notes`	`string`	Descriptive text for the software package.

`patchStatus`	`string`	The state of the patch in its lifecycle. Allowed values:
		`AVAILABLE`
		`DEPRECATED`
		`UNAVAILABLE`
		`UNTESTED`
		(The `UNAVAILABLE` state means the patch has not been uploaded.)

`rebootOnInstall`	`boolean`	Indicates whether to reboot the managed server after successful installation of the software package.
`rebootOnUninstall`	`boolean`	Indicates whether to reboot the managed server after successful uninstallation of the software package.
`removeFlags`	`string`	Optional arguments passed to the uninstaller on the managed server.
`softwarePackageDescription`	`string`	An informational field describing the package.
`softwarePackageDisplayName`	`string`	The name of the software package displayed in the Opsware Command Center.

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`softwarePackageFileName`	`string`	The full name of the file (not including the path) containing the package.
`softwarePackageID`	`decimal`	The unique identifier (primary key) for the `SoftwarePackage` object.
`softwarePackageLocation`	`string`	The fully qualified name of the file (including the path) containing the package. The file resides in the Software Repository.
`softwarePackageName`	`string`	The name of the software packaged used internally by the Opsware System.
`softwarePackageType`	`string`	The type of the package, such as HOTFIX, RPM, MSI, LPP, APAR, ZIP.
`softwarePackageVersion`	`string`	The number of times the software package has been uploaded into the Software Repository.
`softwareRelease`	`string`	For RPMs only, the release info generated by: `rpm -q.`
`softwareVersion`	`string`	For RPM, the version info generated by: `rpm -q`. For MSI, the version specified by the user.
`status`	`string`	Values include: `ACTIVE` `DELETED`

SoftwarePackageFilter

Determines the search results of the Software Package Web Service `getIDs` and `getList` operations.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`customerIDs`	`decimal[]`	An array of IDs of the customers attached to the software packages to retrieve.
`platformIDs`	`decimal[]`	An array of IDs of the platforms attached to the software packages to retrieve. A platform represents an operating system.
`softwarePackageIDs`	`decimal[]`	An array of software package IDs to retrieve.
`softwarePackageName`	`string`	The name of the software package to retrieve.

SoftwarePackageIDOverrideTuple

A tuple in an ordered list that indicates the installation order of inherited software packages. If a software package inherits other software packages in the node hierarchy, and if a server is attached to the node, then the Opsware System must determine the order in which to install these software packages.

Elements

ELEMENT NAME	SOAP DATA TYPE	DESCRIPTION
`overrideType`	`string`	Either a plus symbol (+) or a minus symbol (-). The plus symbol indicates that the software package is inherited and installed on servers attached to this node. The minus symbol means that that the node is aware that its parent has a particular package, but the node does not inherit this package and servers attached to the node do not have this package installed.
`softwarePackageID`	`decimal`	The unique identifier (primary key) of the `SoftwarePackage` object.