

The logo for ZephyrTel is centered within a rectangular box with a diagonal hatched pattern. The text "ZephyrTel" is rendered in a classic serif font, with the 'Z' and 'T' being significantly larger than the other letters. The 'y' has a distinct tail that curves to the left.

ZephyrTel

**Service Gateway 5.1.1.0**

**Release Notes and Installation Instructions**

© 2019 ZephyrTel. All Rights Reserved. These materials are provided by ZephyrTel for informational purposes only, without representation or warranty of any kind, and ZephyrTel shall not be liable for errors or omissions with respect to the materials. The only warranties for ZephyrTel products and services are those set forth in the express warranty statements accompanying such products and services, if any, and nothing herein shall be construed as constituting an additional warranty. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express written permission of ZephyrTel. The information contained herein may be changed without prior notice. Some products marketed by ZephyrTel contain proprietary software components of other software vendors. ZephyrTel and other ZephyrTel products and services referenced herein as well as their respective logos are registered trademarks or trademarks of ZephyrTel or its affiliated companies.

Original filename: Service Gateway 5.1.1.0 Patch Release Notes.pdf

Generated on: 12/03/2019 10:53 AM

# Contents

---

<b>Introduction</b> .....	<b>4</b>
<b>Enhancements</b> .....	<b>5</b>
<b>Issues Addressed</b> .....	<b>6</b>
<b>Patch Contents</b> .....	<b>9</b>
<b>Before Upgrading</b> .....	<b>10</b>
Backups .....	10
<b>Installation Instructions</b> .....	<b>11</b>
Troubleshooting and Manual Installation .....	11
Updated Service Gateway Integration JAR .....	11
New UI Properties for Translation .....	12
<b>Database Changes</b> .....	<b>13</b>
<b>Integration Interface Changes</b> .....	<b>14</b>
<b>Appendix A</b> .....	<b>15</b>
Defining Port Mappings with Parameter Group Templates .....	15
<b>Appendix B</b> .....	<b>20</b>
Reading Port Mappings into Complex Attributes .....	20

## Introduction

---

This document describes the changes included in this patch to Service Gateway, along with the patch installation instructions. This patch can only be applied directly to version 5.1 of Service Gateway. Earlier versions must first be upgraded to 5.1.

## Enhancements

---

The following new features and enhancements are included in this patch:

### **ZenDesk 520883 - Cannot specify TargetFileName in Download RPC**

Service Gateway 5.1 introduced the ability to specify the target file name for a device file, but required using an SQL statement to directly update the database to set that value. In this release, the user interface has been updated to allow a target file name to be specified for a device file, eliminating the need to directly manipulate the database.

### **ZenDesk 520898 - Allow option to group Script File templates**

Script file templates now include a configurable option to group, or concatenate, multiple script file templates together into a single file for delivery to the device. The available grouping options are:

- Send Script File Separately - to send this script file by itself
- Group Script Files across all services - to concatenate all script files templates across all services together
- Group Script Files within a single service - to concatenate all script files within a single service together

The default is to send each script file separately.

When grouping multiple script files together, the filename specified in the Download RPC will be the filename specified for the last template in the list that is grouped together for delivery.

### **ZenDesk 520899 - Deliver multiple port mappings with one template**

Parameter Group Templates have been enhanced with better support for provisioning port mappings on devices. Refer to Appendix A of these release notes for more information on how to use this feature.

### **ZenDesk 1076095 - Reprovision credentials on failed CPE authentication if device sends a CONNECTION REQUEST while there is an RPC queued for that device**

Service Gateway has been updated to automatically reprovision CPE credentials on a device when authentication fails, but only under the following conditions:

- the device sends a CONNECTION REQUEST Inform, and
- there is an RPC queued in server waiting queue for that device

Under these conditions, it can be reasonably assumed that the device is responding to a request from Service Gateway, and that it is safe to reprovision the CPE credentials.

### **ZenDesk 1855318 - Read Port Mappings into a complex attribute**

The CWMP Get Parameter Values (Attribute Storage) policy action has been enhanced to read an entire port mapping definition into a complex attribute, without having to specify all requested parameters or attributes. Refer to Appendix B of these release notes for more information on how to use this feature.

## Issues Addressed

---

The following issues have been addressed by this patch.

### **ZenDesk 520897 - Lots of "dead" records in SPRT\_NC\_CPE\_FW\_UPGRADE**

A new record purging entry has been added to the SPRT\_EC\_RECORD\_PURGING table to purge any in-progress records that have been there 5 minutes longer than the ACS Server CPE File Transfer Timeout value.

### **ZenDesk 522056 - Cannot change device list query for a policy**

Any changes to specify a new device list query as the device filter for an existing policy were not being saved. This has been corrected.

### **ZenDesk 567713 - Workflow times out when device does not setup a session to receive the Download RPC**

Script file templates were using the ACS Server CPE File Transfer timeout, resulting in a several minute delay in several cases, including when the device fails to initiate a session with the ACS. The regular ACS Server Timeout is now used for issuing connection requests, and waiting for the device to initiate the session. This should result in shorter delays when the device does not start a new session.

### **ZenDesk 897271 - Timestamp in SPRT\_EC\_DEVICE\_FW\_HIST is not UTC**

Service Gateway has been updated to ensure that timestamps in the SPRT\_EC\_DEVICE\_FW\_HIST table are now stored in UTC.

### **ZenDesk 1070934 - User account with realm restrictions cannot view Subscriber information**

This was due to an incorrect SQL query when a realm-constrained user tried to view subscriber information. This has been corrected.

### **ZenDesk 1075958 - Server Waiting Queue should be cleared when the device sends a BOOTSTRAP**

Any RPCs that are queued in server waiting queue for a given device are now cleared if the device sends a BOOTSTRAP Inform.

### **ZenDesk 1079068 - Multi-line values in attributes causes validation error in UI**

The regular expression used for validating the attribute values has been updated to accept newline characters.

### **ZenDesk 1079068 - Multi-line values in attributes evaluates to FALSE in conditions**

The regular expression used for evaluating attribute values as part of conditions and expressions has been updated to accept newline characters.

**ZenDesk 1662440 - CallerID is being lost in some circumstances, interrupting workflow.**

The callerId, which is used for notifying the ACS Workflow Engine at the end of an event-driven policy that it can move on to the next step in the workflow, was not being retained when the ACS Workflow itself was checking to see if the current device was queued to be processed in another policy. This was resulting in the premature termination of the workflow when there was still more work to do. The callerId is now retained in that instance, allowing the entire workflow to complete.

**ZenDesk 1832872 - Inform events triggered by a policy workflow, can start a parallel ACS workflow execution flow**

This issue was due to a condition where a device could send an Inform message while Service Gateway was currently processing a previous Inform message from that device. The example that caused this scenario was a device sending a BOOTSTRAP Inform, which results in multiple policies being invoked sequentially to process the event. The first policy included a firmware upgrade, which resulted in the device sending a new Inform message that included the BOOT event code. The timing of that BOOT Inform in this example coincided with the BOOTSTRAP\_COMPLETE workflow state, which resulted in two application servers (one processing the original BOOTSTRAP Inform, and the other processing the new BOOT Inform) both deciding to proceed with the next policy in the list for this device, which happened to be the Configuration policy. This resulted in the Configuration policy being invoked before the Boot policy for the second event. To prevent this scenario from happening, Service Gateway now stores the Inform event identifier in the same database table as the workflow state in order to avoid this rare race condition, and uses that value to determine if another event is already in progress for that device, and queues the incoming event if that is the case, to be processed later.

**ZenDesk 1857754 - Inventory upgrade is not working**

The InitialContext, used for connecting to the application server, was not being initialized properly when trying to update the LAST\_USED timestamp on the SPRT\_EC\_FILE table when performing a firmware upgrade. This has been corrected.

**ZenDesk 1982955 - UI Logout does not really log out**

A JBoss-specific logout function was being included in the WebLogic distribution, resulting in the user information and session not being properly invalidated on logout unless all browser windows were closed. This would allow anyone to simply click on the browser refresh button, or to provide any user credentials, and the previous session would continue as if it had never logged off. The correct logout function is now included in the WebLogic distribution.

**ZenDesk 1985678 - Retried Config Sync only sends subset of templates for device**

A race condition existed when the Synchronize Configuration policy action ran to update the list of templates in the Database of Record, and to then apply those templates to the device. The race condition existed when the first process had not yet committed those changes to the database, resulting in the second process having an incorrect list of templates to apply. This scenario would occur whenever there were already records in the DBoR from a previous failed attempt, such as when a device is offline.

The transaction attributes have been updated to ensure that all changes are committed properly to the database before the next step of the process attempts to apply the templates to the device.

**ZenDesk 1986087 - NullPointerException in PolicyActionAttributesQueryResultMDB**

Additional null checks with warning log messages have been added to report the missing data, allowing the processing to continue without exception stack traces in the logs.

**ZenDesk 2003816 - Missing index for clearing configuration lock**

The identified query was performing a full table scan when trying to release configuration locks due to NULL values in the table. A new function-based index has been added, and the query has been updated to use the function based index.

**ZenDesk 2003908 - Missing index on nc\_request cleanup**

The identified query was performing a full table scan when trying to purge expired ACS requests due to a missing index. The index has been added.

**ZenDesk 2028192 - STUN and XMPP servers lose connection when application server restarts**

The STUN and XMPP servers will now automatically reconnect when an application server restarts.



## Patch Contents

---

In addition to this document, the patch includes the following files:

- packages/servicegateway-cwmp-5.1.1.0.051.tar.gz
- packages/servicegateway-integration.jar
- packages/servicegateway-jboss-5.1.1.0.051.tar.gz
- packages/servicegateway-openfire-5.1.1.0.051.tar.gz
- packages/servicegateway-stun-5.1.1.0.051.tar.gz
- packages/servicegateway-timer-5.1.1.0.051.tar.gz
- packages/servicegateway-weblogic-5.1.1.0.051.tar.gz
- packages/sprtWeblogicSecurityProviders.jar
- patch/sql/OracleDBChangesFrom5.1.0.0\_DDL.sql
- patch/sql/OracleDBChangesFrom5.1.0.0\_DML.sql
- patch/NewUIProperties.txt

These files will be updated in various locations to address the issues described above.

This patch updates the Service Gateway EAR file. Once this patch is applied to Service Gateway 5.1, the Service Gateway EAR file version will be:

- ServiceGateway 5.1.1.0.051

Additionally, the patch also includes the following directories which contain files required by the installer itself:

- bin
- configuration
- etc
- interface
- lib
- scripts
- sql
- utils

## **Before Upgrading**

---

### **Backups**

Regular backups of the database and file systems should be performed prior to performing an upgrade.

## Installation Instructions

---

This patch is packaged with a web-based installer to ease the upgrade process. The installer is used in the same way as during a new install.

It is necessary to run the installer on all servers that make up the Service Gateway installation. This is so that the installer can update, at minimum, patch level information on each server.

Prior to upgrading any server, the ACS servers must be shut down and all user interface and EAI activity must cease.

Perform the upgrades to the Application Servers and Database Schema first. When patching a WebLogic clustered environment, the Admin server must be patched before any managed servers are patched. WebLogic servers must be running before the patch process can be started. Ensure that the WebLogic configuration is not locked.

Once the application servers and database schema have been successfully upgraded, proceed with the upgrade of each ACS server. The ACS upgrade will automatically restart the ACS.

The steps to upgrade each server are as follows:

1. Extract the contents of the patch archive to a temporary location.
2. Copy the JDBC driver for the database to the root directory of the extracted patch contents.
3. Enter the "bin" directory and start the run script appropriate to the operating system. run.bat for Windows, and run.sh for Solaris or Linux.
4. Under Windows, the default web browser is automatically launched and directed at <http://localhost:8888/>. On Solaris or Linux, a web browser must be launched from any computer on the network and directed at the installation site manually. The installer listens for HTTP connections on port 8888 of the server the installer is running on.
5. After accessing the installer web UI, select "Update an existing instance" and click "Next".
6. Once the target instance has been selected and the license agreement has been accepted, the patch prerequisite scripts will run. If they are all successful, clicking "Next" will start the upgrade process.

## Troubleshooting and Manual Installation

If the installer fails for any reason, installer.log should be backed up to a safe location so that there is no loss of information needed to diagnose the problem and understand the current state of the application. This file should be sent to Apteian Technical Support for review. Manual patch and rollback instructions are available to Apteian technicians to assist in recovery from a failed upgrade.

## Updated Service Gateway Integration JAR

The patch contains a new copy of servicegateway-integration.jar, which is used by all utilities that interface with Service Gateway. Any custom code or integration applications must be updated to use the new integration jar file.

## **New UI Properties for Translation**

New UI properties have been added. These are already present in the EAR file that has been deployed, so no further action is required. However, if any translations have been created for the installation, the new tokens will need to be translated and added to the translated properties files. A list of the new tokens can be found in the NewUIProperties.txt file included with the upgrade package.

## Database Changes

---

A new column, AGGREGATION\_TYPE, has been added to the SPRT\_SG\_TCP\_CWMPTEMPLATE\_I table.

A new column, EVENT\_IDENTIFIER, has been added to the SPRT\_NC\_CPE\_WF\_STATE table.

A new table, SPRT\_SG\_TCP\_CWMPTEMPLATE\_QUEUE, has been added to handle the grouping of Script File templates.

Two new indexes, SPRT\_EC\_DEVICE\_IDX101 on SPRT\_EC\_DEVICE and IX\_SPRT\_NC\_REQUEST\_3 on SPRT\_NC\_REQUEST, have been added to help performance.

## **Integration Interface Changes**

---

There are no changes to the EAI Web Services in this release.

## Appendix A

### Defining Port Mappings with Parameter Group Templates

Three of the operation types for a Parameter Group template have been enhanced for better handling of more complex configurations such as port mappings. This enhancement takes advantage of the existing "Repeat for Group" Velocity script directive that references complex attributes.

The power behind the "Repeat for Group" directive is that it allows you to define a number of complex attributes on a device, and this will result in a repeated set of RPC commands that get delivered to the device to set the parameters in question. For example, say you wanted to define 3 specific port mappings on a device. You could define a complex attribute group, with the attribute names needed to specify the data you would want to set on the device.

The way the "Repeat for Group" directive was originally designed to work was to specify a parameter you wanted to set on the device. For example, say you wanted to set *InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.Alias*. You would select that as the parameter to set. For the index values you would specify "Require Index", and specify an index value of "#" (without the quotes). Then in the value to set, you would use the Velocity Script editor to select the "Repeat for Group" directive, and select the complex attribute group. Following the instructions in the comment text that gets inserted into your script when you select that option, you can then specify how the value would get set.

In the first iteration of the "Repeat for Group" implementation, the index values of the complex attribute would be used as the index values in the parameter name. So if you had three instances on the complex attribute defined on your device, with index values 1, 9, and 12, you would end up with 3 parameters included in the SetParameterValues RPC that gets sent to the device, with index values 1, 9, and 12, and values as determined by the Velocity script.

In this release the "Repeat for Group" directive has been enhanced for more powerful handling. There are two main enhancements. The first is that it has been extended to include the Add Object and the Delete Object operation types. Previously support was only available for the Set Parameter operation. And secondly, the first iteration only allowed you to use the attribute index to map to the parameter index. This release includes an enhancement to specify an attribute that names the cached index value to use. More on that later.

Let us use an example to illustrate how the "Repeat for Group" directive works, using the attribute index to map to the parameter index. Here is our Parameter Group Template definition:

Current Commands			Delete
Operation	Object	Value	
Set Parameter	InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.Alias	## This special directive w	
	1. Prefer index: # #		
Set Parameter	InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.DestIPAddress	## This special directive w	
	1. Require index: #		

And here is the content of the Value field for one of the parameters:

```
## This special directive will cause a script to be evaluated multiple
## time, once for each instance of an attribute group present on the
## specified object type.
##
## This directive may be placed in a comment anywhere in the script.
##
## If the specified attribute group is not associated with the object,
## the script will not be evaluated at all.
##
## This directive enables the following variables which will contain
## different values for each iteration:
##     Current Attribute Group:      $currentGroup
##     Current Attribute Group Index: $currentGroupIndex
##     Attribute value:              $currentGroup.AttributeName
##
## REPEAT FOR EACH device.FORWARDING
$currentGroup.ALIAS
```

Everything that appears as a comment is inserted when you select the "Repeat for Group" directive when adding an attribute from the Script Editor drop-down. The important line that has to be there is the last comment line: **## REPEAT FOR EACH device.FORWARDING**.

This tells the script compiler that this is a "Repeat for Group" directive, that the attribute group belongs to a device, and that the attribute group name is "FORWARDING". Everything below that line is the velocity script that evaluates to the value for each attribute instance associated with the device. You will notice that the attribute value is referenced as **\$currentGroup.ALIAS**, which follows the syntax described in the comment block.

Let us now turn our attention to the device definition, in particular the attributes associated with the device, as shown below:

Current Attributes <span style="float: right;">Delete</span>	
Name	Value
FORWARDING.16	
ALIAS	Alias16
DEST_IP_ADDRESS	1.2.3.16
FORWARDING.21	
ALIAS	Alias21
DEST_IP_ADDRESS	1.2.3.21



When you associate that template with the device and synchronize the configuration, the following list of parameters get sent to the device in the SetParameterValues RPC:

```
<ParameterValueStruct>
  <Name>InternetGatewayDevice.Layer3Forwarding.Forwarding.16.Alias</Name>
  <Value xsi:type="xsd:string">Alias16</Value>
</ParameterValueStruct>
<ParameterValueStruct>
  <Name>InternetGatewayDevice.Layer3Forwarding.Forwarding.21.Alias</Name>
  <Value xsi:type="xsd:string">Alias21</Value>
</ParameterValueStruct>
<ParameterValueStruct>
  <Name>InternetGatewayDevice.Layer3Forwarding.Forwarding.16.DestIPAddress</Name>
  <Value xsi:type="xsd:string">1.2.3.16</Value>
</ParameterValueStruct>
<ParameterValueStruct>
  <Name>InternetGatewayDevice.Layer3Forwarding.Forwarding.21.DestIPAddress</Name>
  <Value xsi:type="xsd:string">1.2.3.21</Value>
</ParameterValueStruct>
```

You will notice that the parameter indices and values coincide with the attributes associated with the device.

The addition of support for the Add Object and Delete Object commands means that you can now also use the "Repeat for Group" directive for those commands as well, by specifying the (#) hash mark for the index value and also the index name. This means that you can now add and delete objects from the data model from a single template, using attribute associations on the device to determine the number of and which nodes to add or delete.

Now let us examine the case where you can use an attribute to specify the index name. This allows you to use something other than the attribute group index when associating attribute instances with individual parameters in the data model. With this approach you are not actually specifying the name of the index, but rather an attribute within the group that contains the name of the index. Perhaps a bit of background is helpful here. Whenever you define a Parameter Group Template that includes parameters with indices, there is an option to specify an index name. If you do not provide one, Service Gateway will generate one for you. This index name, along with the actual index value, gets stored in the database in a table called SPRT\_SG\_CWMP\_ICACHE. This happens the first time the Parameter Group Template is processed. Once those values are in the database, then the Parameter Group Template will use those values each time it gets processed.

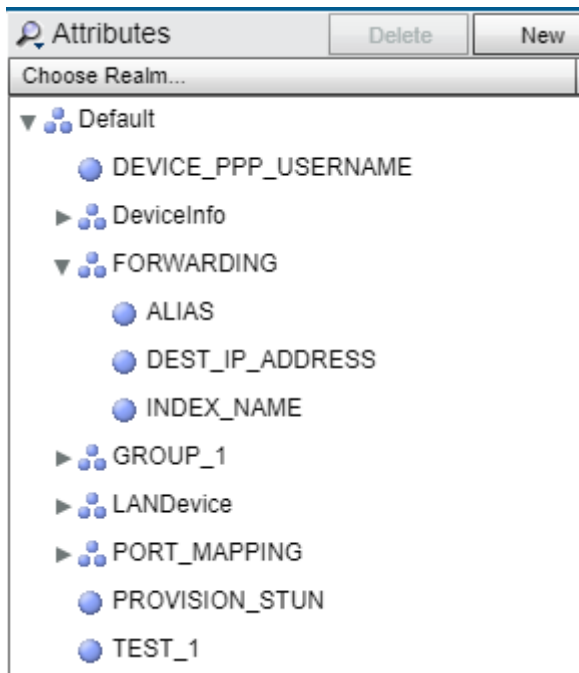
One place where this is beneficial is in the following example:

Current Commands	
Operation	Object
Set Parameter ▼	InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.Alias 1. Prefer index: ▼ 5 Idx_001
Set Parameter ▼	InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.DestIPAddress 1. Require index: ▼ Idx_001

In this example, the SPRT\_SG\_CWMP\_ICACHE table will contain a record with an index name of Idx\_001, and it will be associated with the index value that applies to the first parameter. Here we specified the "Prefer Index" option, meaning that our preference is to use the specified index value

of 5, provided it exists on the device. If it does not exist, the AddObject RPC will be invoked to add a new row to the table in the data model, and whatever index value that returns will be associated with the index name stored in the database. Then, the second parameter in the Parameter Group Template, which specifies the index using that index name, will be set to the same index value as the first parameter.



OK, now to the new option where you can reference the index name using an attribute. In this case you would define an attribute on your group that is used to store the attribute name, as shown below:



In this case the attribute called INDEX\_NAME would hold the name of the index that will be referenced in the Parameter Group Template using the "Repeat for Group" directive. On the device we would need to include that attribute on the device configuration, as shown below:

Current Attributes <span style="float: right;">Delete</span>	
Name	Value
FORWARDING.[16]	
ALIAS	Alias16
DEST_IP_ADDRESS	1.2.3.16
INDEX_NAME	Index 16
FORWARDING.[21]	
ALIAS	Alias21
DEST_IP_ADDRESS	1.2.3.21
INDEX_NAME	Index 21

Now let us examine the Parameter Group Template definition where we would specify this attribute. You don't have to specify the group name, just the attribute name. The group name is taken from the contents of the "Repeat for Group" directive in the Velocity Script in the Value field. This is illustrated below:

Current Commands <span style="float: right;">Delete</span>		
Operation	Object	Value
Set Parameter ▼	InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.Alias	## This special directive w 
	1. Prefer index: ▼ #INDEX_NAME #INDEX_NAME	
Set Parameter ▼	InternetGatewayDevice.Layer3Forwarding.Forwarding.{i}.DestIPAddress	## This special directive w 
	1. Require index: ▼ #INDEX_NAME <input type="text"/>	

The way this works is very similar to the way it worked when only the (#) hash mark was specified, except rather than using the attribute index value for the index name and index value, it uses the value from that attribute instance for the index name. In our example above, the index names that would be stored in the SPRT\_SG\_CWMP\_ICACHE table would be "Index\_16" and "Index\_21". In a similar manner, the Add Object and Delete Object can also reference the index attribute in order to determine the index name. When specifying the index names in this manner, the attribute index values are not used when processing the templates.

## Appendix B

### Reading Port Mappings into Complex Attributes

The CWMP Get Parameter Values (Attribute Storage) policy action has been enhanced for better handling when dealing with nodes and tables in the data model. Previously when using this policy action you had to define a 1-to-1 mapping between each individual parameter in the data model, and the attribute you wanted to store it in. This included specifying the exact indices for both.

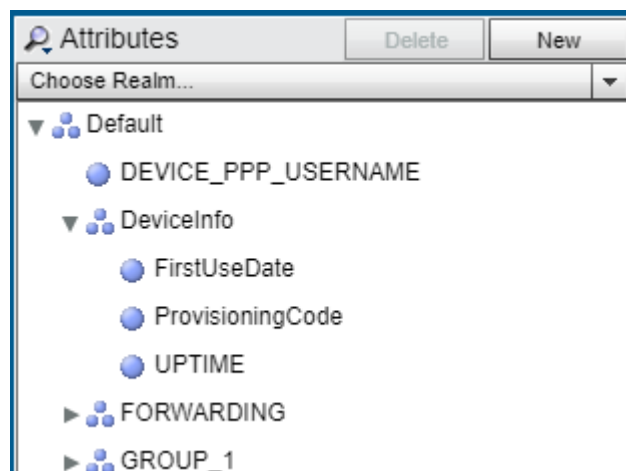
Beginning with this release you will be able to specify a node in the data model, and the complex attribute group that you want to store those values in. You can also specify wildcards for the indices in order to retrieve and store an entire table. When working with parameter that contain more than one index value, only the last index can be a wildcard value. All other indices in the parameter name must to specified explicitly.

**Note:** Due to a way that the attribute tree is generated in the user interface, it is not currently possible to select the group name of an attribute. Instead, you need to select an attribute that is within that group. It does not matter which attribute you select, as the code simply uses the group name. This will be addressed in a future release so that the group itself can be selected.

A requirement for this feature is that the names of the complex attributes (not including the group name) has to match the names of the parameters in the data model (not including the full path to that parameter name). For example, let's say we wanted to read some of the values from the *InternetGatewayDevice.DeviceInfo*. node into a complex attribute, and the parameters we wanted to read into the attributes are:

- FirstUseDate
- ProvisioningCode
- UpTime

So we could define a complex attribute such as the following:



Note that the spelling of the complex attribute names must match the spelling of the parameter names, but are not case sensitive. So UPTIME and UpTime will match.

When configuring the policy action, you would select the *InternetGatewayDevice.DeviceInfo*. node as the "parameter", and any attribute from within the target group. Specify whatever index value you want to use, as shown in the image below:

Configuration	Conditions	Workflow Operations	Retries and Timeouts
---------------	------------	---------------------	----------------------

---

General	Parameter Mappings
---------	--------------------

Parameter Name	Attribute Name
InternetGatewayDevice.DeviceInfo.	DeviceInfo.6.ProvisioningCode

Parameter Name

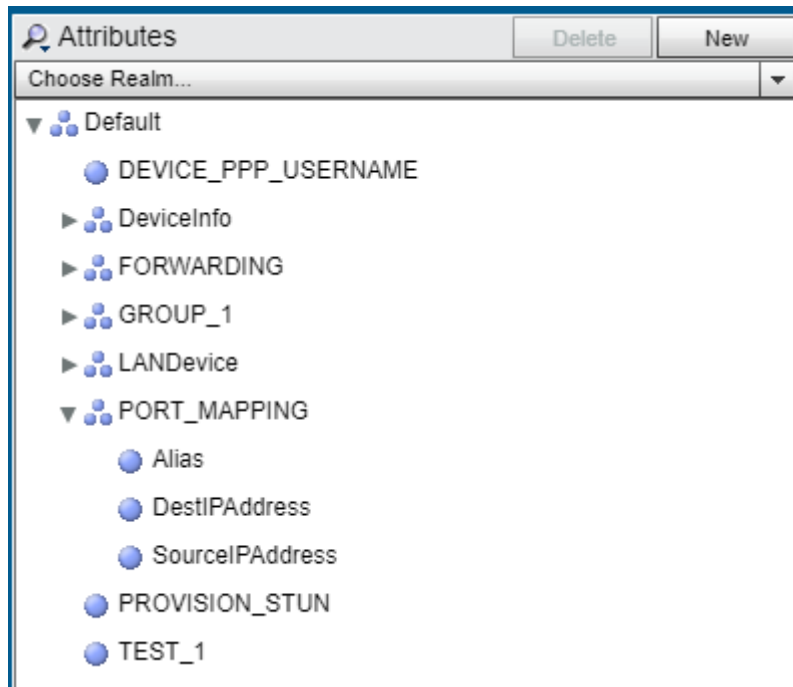
Attribute Name

When the policy action runs, it will retrieve all parameter values that are under the *InternetGatewayDevice.DeviceInfo.* node, but it will only store the values that have a corresponding attribute name (not case-sensitive) in the complex attribute group. Whatever index value that you specified in the policy action for the complex attribute will be used when storing the values on the device. So in the example above, the complex attribute would get an index value of 6.

This feature can also be used for tables in the data model. Suppose we wanted to read some port mapping information into a complex attribute, and that we want the following parameters that can be found under *InternetGatewayDevice.Layer3Forwarding.Forwarding.{j}.*:

- Alias
- DestIPAddress
- SourceIPAddress

We could define a complex attribute such as the following:



When configuring the policy action you would specify the *InternetGatewayDevice.Layer3Forwarding.Forwarding* node as the "parameter", and any attribute from within the target group. You could specify a specific index value for the parameter, and the policy action will only retrieve that one row from the table in the data model. In that case the example is exactly the same as described above when retrieving the DeviceInfo node.

Alternatively you could specify a wildcard (\*) for the index, as shown in the image below, then the policy action will return all rows for that table in the data model, and it will create one complex attribute entry for each row that is returned, using the index values from the returned rows as the index values for the complex attributes. So if the data model contained 3 rows with index values 1, 3, and 27, then the end result would be 3 complex attributes on the device with index values 1, 3, and 27, matching the rows returned from the device.

Configuration		Conditions	Workflow Operations	Retries and Timeouts
General		Parameter Mappings		
				Delete
Parameter Name				Attribute Name
InternetGatewayDevice.Layer3Forwarding.Forwarding.*				PORT_MAPPING.0.Alias
Parameter Name	<input type="text"/>			
Attribute Name	<input type="text"/>			
	Add			

**Note:** Even though the above image shows an index value of 0 for the selected attribute, that will be ignored. The presence of the wildcard for an index in the parameter name indicates that a table retrieval is being performed, and the resulting attribute index values will match the parameter index values in the device's data model. The presentation of this information will be addressed in a future release.