Xanadu IT Operations Management

Last updated: August 5, 2024

Some examples and graphics depicted herein are provided for illustration only. No real association or connection to ServiceNow products or services is intended or should be inferred.

This PDF was created from content on docs.servicenow.com. The web site is updated frequently. For the most current ServiceNow product documentation, go to docs.servicenow.com.

Company Headquarters
2225 Lawson Lane
Santa Clara, CA 95054
United States
(408)501-8550

Create or customize patterns

Create or modify a discovery pattern and define its basic attributes.

Before you begin

Make sure that the application for which you want to create a pattern, has a corresponding configuration item (CI) type and a CI classification. If the CI type you require is not in the list, create it as described in Create CI types for Service Mapping and Discovery.

If your ServiceNow instance uses domain separation and you have access to the global domain, log in to the relevant domain. The selected domain must be a domain without any child domains.

Role required: pd_admin

Basic knowledge of programming is desirable.

About this task

Patterns can be of the infrastructure or application type. Infrastructure patterns are used only by Discovery for creating lists of devices. Application patterns serve both Service Mapping and Discovery, which use the same application patterns for their purposes.

Procedure

1. Navigate to **All > Pattern Designer > Discovery Patterns**.
2. Click **New** or select the relevant pattern from the list.
3. Define the basic pattern attributes on the **Basic** tab.

Field	Description
Pattern type	• Select **Application** for an application pattern. It can be used both for top-down discovery performed by Service

Field	Description
	Mapping and horizontal discovery performed by Discovery. • Select **Infrastructure** for an infrastructure pattern used for the horizontal host discovery performed by Discovery.
Name	Enter the pattern name. This name must be unique to this pattern. Use self-explanatory names such as `Apache on Unix pattern`.
CI type	Select the CI type which you want this pattern to discover. **Note:** Discovery can find multiple CIs that belong to the same CI type.
Operating system [Application patterns only]	Select the operating system that the selected CI runs: • Click **All** if the CI runs on more than one operating system. Or • Select the relevant operating systems from the list.
Run Order [Application patterns only]	For an application pattern used by Service Mapping, define when the pattern runs a. Select the order in which this pattern always runs:

Field	Description
	• **Before** • **After** b. Then select the other applicable pattern. This field is only relevant if a particular pattern can be confused with another pattern. For example, both IIS and MS Exchange applications have an HTTP entry point. However MS Exchange uses some of the components of IIS. Therefore, if the IIS pattern ran first, discovery might incorrectly identify MS Exchange as IIS. To prevent this error, in the **Run Order** field in the MS Exchange pattern definition, select **Before** and **IIS**.
Description	Provide a description for this pattern.

4. (Optional) When creating an application pattern, make the MID Server run this pattern only if the process identified on a CI matches the classification criteria for this pattern, select **Enforce Process Classification**.
   All simplified patterns created from generic applications, have this attribute enabled. For more information about creating process classification, see Discovery classifiers.

5. Define a set of identification steps for every incoming connection of a configuration item (CI).

   a. In the **Identification Section**, click **New** and then configure the following parameters.

Field	Value
Name	Unique name for the identification section.
Entry Point Types [Application patterns only]	Select all relevant entry point types. Every CI has incoming connections that are referred to in Pattern Designer as entry points. You base your CI identification process on the CI entry points creating steps and defining step operations and variables for every entry point separately. For application patterns used by Discovery, enter either TCP or All.
Find Process Strategy [Application patterns only]	Select the appropriate strategy for finding the process that populates the Process variable in the Temporary Variables table. • **Listening Port**: The entry point is the listening port. • **Target Port and IP**: The entry point port communicates with another server. • **None**: The process variable is not populated.

Field	Value
	Typically, the port type of the entry point determines the strategy.
Order	Select a number that determines the order in which Service Mapping or Discovery use identification sections. The section with the lowest order number is used first.

b. Click **Save**.

c. Click the identification section name.

d. Define discovery steps as described in Define discovery steps.

e. Click **Save**.

6. If necessary, create more identification sections.
7. Save the pattern.
8. Upload the updated set of patterns onto the MID Server:
   a. Navigate to **Discovery > MID Servers**.
   b. Alternatively, navigate to **Service Mapping > MID Servers**.
   c. Click **Pattern Sync to Mid**.

What to do next

For application type patterns, continue with Define the connection section.

- Example of creating an application pattern

  Follow this example to see a step-by-step process of creating and defining the identification section for a new application pattern.

Related concepts

- Exploring Cloud Discovery

Related topics

- KB0747679: Pre/Post Processing Scripts for patterns

Define discovery steps

For each Identification Section and Connection Section entry that you added to the discovery pattern, define discovery steps. These steps are the basis for discovery.

Before you begin

Role required: pd_admin

About this task

You can choose to define the discovery steps immediately after adding an Identification, Extension, or Connection section, or you can choose to do it after adding the required sections.

Define an operation for every step. The type of operation dictates which parameters and variables need to be configured.

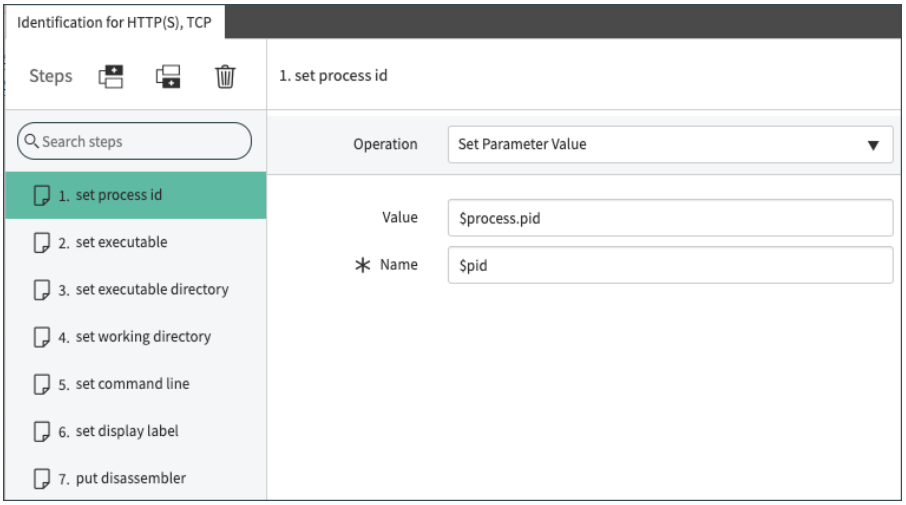
Procedure

1. On the pattern form, click the relevant entry in the **Identification Sections** or **Connectivity Sections**. Connectivity Sections is for Service Mapping only.

   The Pattern Designer opens showing the Steps tree on the left.

   If no discovery steps have been identified for this pattern, the **Untitled Step** appears in the Steps tree in the left pane of the window.

2. (Optional) When modifying an existing pattern, search the Steps tree to find the relevant step.



3. To add a new step:
   - Click to add a step above the current step.
   - Click to add a step below the current step.
4. Select an operation from the list and then fill in the fields that appear for the operation.

Operation	Objective
Change User	Use operating system credentials instead of the default administrative credentials.
Create Connection	Provide information about outgoing connections. This is for the connectivity section of a pattern that applies to Service Mapping only.
Filter Table	Filter a table according to specified criteria.

Operation	Objective
Find Matching URL	Find the best match for a URL in a list of URLs.
Get Registry Key	Query for registry keys.
Get Process	Search for a process according to specified criteria.
Library Reference	Combine a number of steps to be executed as a group.
Match	Match a condition and stop running the pattern if a condition is not met.
Merge Table	Merge two tables.
Parse Command Output	Extract information from the output of the command.
Parse File	Extract information from a file.
Parse a URL	Break down a URL into its components.
Parse Variable	Extract information from a variable.
Define an HTTP Get Call query	Extract information from devices using HTTP protocol.
Cloud Rest Call	Extract information from cloud computing devices like Microsoft Azure or Amazon Web Services
Create Relation/Reference	Create relationships and references between CIs that

Operation	Objective
	were discovered within the pattern.
Put File	Transfer a file to a remote system.
Run an SSH script file	Run composite commands or sequences of commands on Unix-based hosts.
Set a parameter value	Set the value of a parameter.
SNMP Query	Execute an SNMP query.
Transform Table	Add computed columns to an existing table.
Unchange User	Switch back to the default administrative credentials.
Union Tables	Append two tables that share the same format.
WMI Method Invocation	Execute a method using WMI (Windows Management Instrumentation).
WMI Query	Execute a WMI query.

If there are custom operations, which do not come with the base system, you can select a custom operation. For more information, see Customize pattern operations.

5. Specify the following discovery step settings.

Field	Description
Precondition	Select this check box to add a specific criteria to the step. If the step is always performed as defined, leave this setting unchecked. For more than one condition, consider defining a step for each condition rather than multiple conditions. For more information, see Make a step conditional.
CI Attributes	Table that is automatically populated with CI attribute variables that are generated when you add a CI type. This table does not support Container or Tabular variables. You can use shortcuts to enter values as described in Enter values and variables in patterns. All variables are notated with a $ prefix and constants are formatted within double quotes. For more information, see Pattern variables.
Temporary Variables	Table that is automatically populated with temporary variables that are generated when you define a discovery pattern.

Field	Description
	You can add or remove variables from the Temporary Variables table. You can use shortcuts to enter values as described in Enter values and variables in patterns. All variables are notated with a $ prefix and constants are formatted within double quotes. For more information, see Pattern variables.

6. To add comments to any step, click the comment icon (), add the text in the comment text and click **Post**.
   The comment icon changes to indicate that there is a comment associated with this pattern step: .
7. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.
8. To exclude the step from the pattern without deleting it, clear the **Active** check box.
   The MID Server skips this step while running the pattern.
9. To delete a step from the section, select the step and click the trash can icon.
10. After you define all steps, click **Save**.
11. On the pattern record, click **Activate** to make that pattern available for use.

What to do next

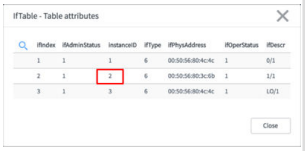
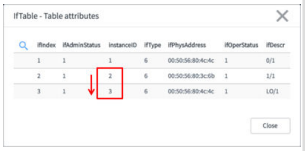
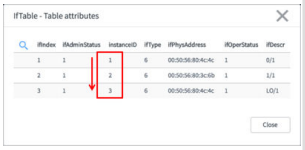
Click **Debug** to access the additional actions and to browse to and open source files rather than looking them up separately.

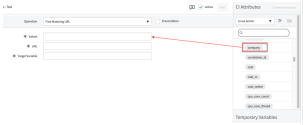
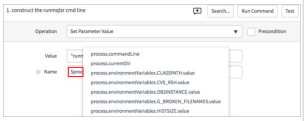
Enter values and variables in patterns

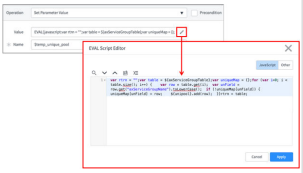
There are several ways to enter values in Pattern Designer. You can use the following types of values in patterns: strings, variables, concatenated variables, and eval() functions. Do not use GlideRecords in patterns.

Useful shortcuts in Pattern Designer

To enter this value	Follow these steps	Example
Enter constant values, a string	Enclose the value in quote marks (").	Enter the path with quote marks (") at the beginning and at the end of the string. 
Enter a value that can change, a variable	Start the variable name with the dollar sign ($).	Enter the variable name beginning with the dollar sign ($). 
Enter a value that can change from a variable that contains several variables	Enter the variable name in the following format: $<container_variable>.<variable>	Enter the dollar sign ($), followed by `process` for the container variable name, period (.), and then `pid` for the name of the string variable. 
Enter a value from the specific field in a table	Enter the variable in the following format: $tabular_variable[row number].column_name	To use the value from the second row in the instanceID column from the IfTable variable, enter

To enter this value	Follow these steps	Example
		`$IfTable[2].InstanceID`. 
Enter values from a specific column in a table sequentially, starting from the current row	Enter the variable in this format: $tabular_variable[].column_name	To use the value from the current row in the instanceID column from the IfTable variable, enter `$IfTable[].InstanceID`. 
Enter values from a specific column in a table sequentially, starting from the first row	Enter the variable in this format: $tabular_variable[*].column_name	To use the value from the first row in the instanceID column from the IfTable variable, enter `$IfTable[*].InstanceID`. 

To enter this value	Follow these steps	Example
		Note: When used in Match a condition with the **Is Not Empty operator**, the system extracts values from fields, even if some fields are empty.
Copy a value or a variable into a field	Select the variable or the value you want to copy from the **CI Attribute** pane and drag it into the target field.	Drag the **company** variable from the **CI Attributes** pane into the Values field. 
Enter a variable using auto-complete	1. Type the dollar character ($) and the first letters of the variable name. 2. Select the relevant value from the list showing all currently available values that match the characters you entered. If only one choice fits, that value is automatically entered into the field.	Variables have a $ prefix. Typing `$P` in a field displays a list of possible values beginning with "P". 

To enter this value	Follow these steps	Example
Specify complex (concatenated) values in fields	Enter a value, then add a plus sign (+), and then enter another value. **Note:** You can drag values and variables to create complex values.	To specify the path, use the install_directory variable and extract of the actual path connected by a plus sign (+). 
Use the eval() function to evaluate a string in the field	1. Type EVAL(and click the edit button  next to the field. 2. In the EVAL Script Editor window, ensure that the JavaScript mode is on. Alternatively, click **Other** for other types of scripts. 3. Type the script in the editor pane. 4. If necessary, use the **Search**, **Replace**, and **Format** buttons to modify the script. Linting and highlighting is available for JavaScript.	To use the eval() function for JavaScript serving as a parameter value, enter the script in the EVAL Script Editor window. 

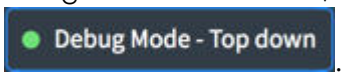
To enter this value	Follow these steps	Example
	5. Click **Apply**.	

Activate pattern Debug mode

Working in Debug mode, Pattern Designer performs all operations as you configure them. It allows you to see results immediately.

Before you begin

Role required: pd_admin

Before starting this procedure, verify that the Debug mode is not activated. If the Debug mode is activated, the debug button appears with a green dot: 

.

Basic knowledge of programming is desirable.

About this task

Many operations require that you enter a specific value from a specific source, for example, a particular value, or a particular location and delimiter, in a particular file. If you are not in Debug mode, you must type details such as path and file name, or an actual relevant value or related information. However, if you work in Debug mode, you can browse to and open information sources such as files, and then select specific values from those files.

Service Mapping and Discovery share a set of preconfigured patterns that cover most of the commonly used devices and applications. Patterns can be of the infrastructure or application type. Infrastructure patterns are used only by Discovery for creating lists of devices. Application patterns serve both Service Mapping and Discovery, which use the same application patterns for their purposes.

Service Mapping and Discovery share patterns, but execute them differently. Discovery runs infrastructure, application, and cloud resource patterns for horizontal discovery, while Service Mapping runs application patterns for the top-down discovery. When you activate Debug mode,

you choose how pattern steps are run: as if by Discovery or as if by Service Mapping.

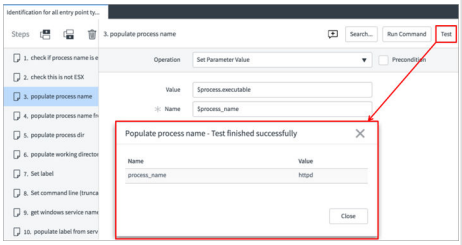
When you activate the Debug mode, Pattern Designer performs an operation in a pattern step and displays discovered values in the Temporary Variables pane.

> **Note:** Debug mode for Windows, SNMP, and Linux Server patterns is supported on IPv6 Host IP.

When you activate the Debug mode, the following additional actions become available:

Debug Mode Actions

Action	Allows you to
Search assistant	Search within files or the registry. **Note:** Perform additional configuration to enable this feature as described in Configure Search Assistant for Windows.
Command Prompt	Check details of file structure, setup, or parameters and values inside specific files on a host before using this information in step operations.
Test	Test the current step and populates variables with discovered values.

Action	Allows you to
	Test result window shows discovered values 

Activate the Debug mode for each browser window or tab separately. For example, if you have the same pattern section open in two or more tabs, you must activate the Debug mode in each tab.

Discovery and Service Mapping use algorithms to select a MID Server for discovery. While working in Debug mode, you can either rely on the MID Server chosen by the algorithm or select a different MID Server from the list. This list contains only MID Servers that are valid, running, and having the matching IP address range for the pattern step. Discovery and Service Mapping use the same MID Server for the entire Debug session, whether the algorithm selected this MID Server or you chose it.

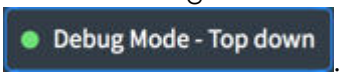
Note:

MID Server auto-selection does not work for IPv6 Host IP in Debug Mode.

Procedure

1. In the Pattern Designer, click **Debug Mode**.
   The Debug Identification Section window is displayed.
2. Fill in the required details for the MID Server entry point:

Field	Description
Select MID Server	(Optional) Select the MID Server for this Debug session.
Debug Type	Select **Top down** for performing top-down discovery with Service Mapping or **Horizontal** for performing horizontal discovery using Discovery.
Select Entry Point Type	Select the entry point type for the discovered CI. Entry point parameters depend on the type you select.
URL	Enter the URL of the CI you are discovering.
Host Name	Enter the host name of the server hosting the discovered application.
Comments	Any considerations or notes.
Use original IP	Select the check box and enter the IP for this entry point.

3. The Debug Identification Section window, click **Connect**.
   The Debug mode is activated. The **Debug** button shows the green dot. For connection sections the Debug Mode button also shows the Debug type you chose: 

.

   When you select a different section in the same pattern, the Debug mode stays activated if it can use the entry point you defined for the previous step.

4. (Optional) If you click another section, to which Pattern Designer cannot connect using the entry point for the previous step, define the entry point again as described in step 2.
5. (Optional) To activate the Debug mode for the same pattern section to which you navigated from the pattern fmorm in a separate browser window or tab, define the entry point again as described in step 2.
6. To deactivate the Debug mode, click the **Debug Mode** button.

Make a step conditional

If necessary, create a precondition that defines how Discovery and Service Mapping execute a pattern step.

Before you begin

Role required: pd_admin

Basic knowledge of programming is desirable.

Create a pattern or select a pattern that you want to modify.

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

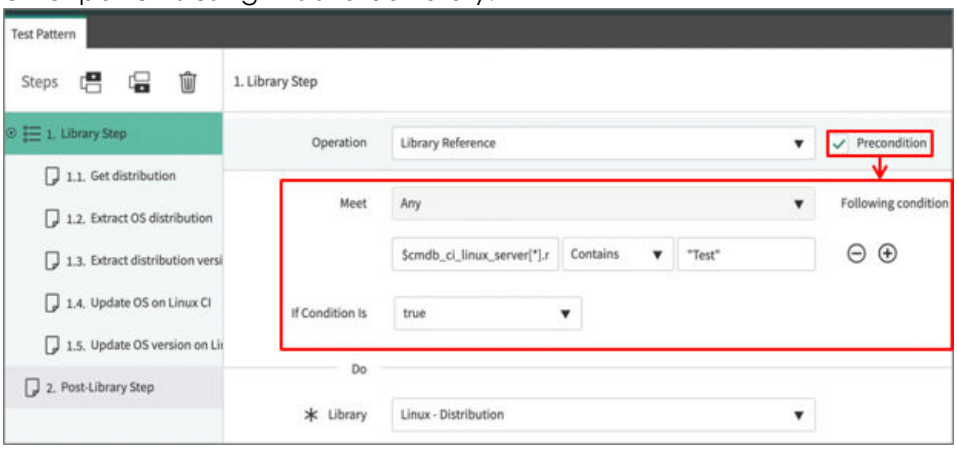
2. Select the relevant pattern step or click  to add a step.

About this task

You can define a single or multiple prerequisite conditions. For example, you can define that a step is run only during the horizontal or top-down discovery using the pattern_runtime_mode variable.

If a precondition defined for a shared library step is not fulfilled, the pattern skips all steps belonging to this shared library. For example, if the

file, to which the variable points, does not contain "test", the pattern does not run the shared library steps 1-1 to 1-5. The precondition exists only for the pattern within which you created it. The precondition does not affect other patterns using this shared library.



Procedure

1. Click the **Precondition** check box next to the **Operation** list.
   Not all operations allow you to use preconditions.

2. In the first condition field, enter the required value.
   For example, enter the actual string or a variable name.

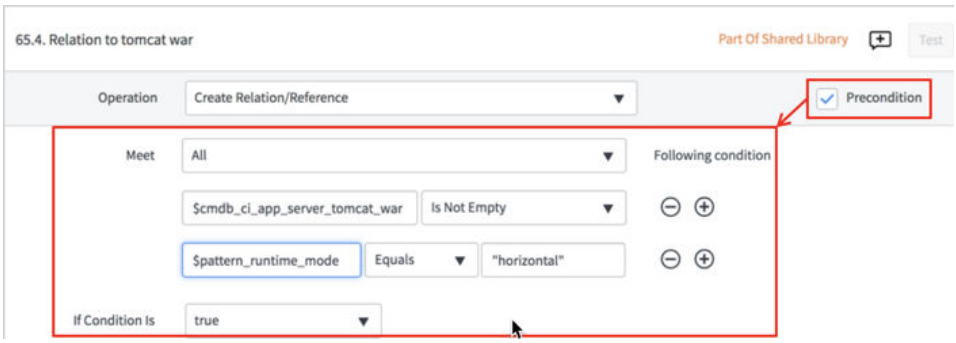
3. Select an operator from the list.
   If you select `Is Empty`, the second field is rendered irrelevant and disappears.

4. To add more conditions, click the plus icon and define the criteria.

5. If you create multiple conditions, define if this pattern must meet all or any of them: From the **Meet** list, select `All` or `Any`.
6. Define if the criteria must be satisfied or not for the step operations to run: From the **If Condition is** list, select `True` or `False`.

Example

You create a pattern step to filter data in the cluster string table to receive a table containing only cluster names. You can set a precondition to filter the table only if the cluster string variable is not empty.



Related tasks

- Create or customize patterns

Customize pattern operations

As part of creating or modifying a discovery pattern, modify existing custom operations that come as part of the base system or add new ones.

Before you begin

Role required: pd_admin or pd_mid

Practical knowledge of Java scripting is required.

About this task

For standard pattern operations, you can define only input parameters. However, there are custom operations for which you can define the business logic and the mechanism of the operation itself in addition to input parameters. For example, you can make a parameter mandatory or define which input parameters to use.

The following custom operations are available:

Cloud REST Call

Extracts information from configuration items of the PaaS (Platform-as-a-Service) type, such as Microsoft Azure or Amazon Web Services. This Java-based custom operation is part of the base system.

HTTP Get Call

Extracts information from configuration items (CIs), which use the HTTP protocol. This custom operation is part of the base system.

Cloud REST Query

Extracts information from configuration items of the PaaS (Platform-as-a-Service) type, such as Microsoft Azure or Amazon Web Services. This Java-script based custom operation is available only after downloading patterns version 1.0.24 or later from ServiceNow Store. Use this operation instead of the Cloud REST Call operation.

In addition to these custom operations, you can create your own operations to serve the needs of your discovery process. Custom operations created by you appear in the list of operations along with operations, which are part of the base system.

Procedure

1. Navigate to **All > Pattern Designer > Custom Operations**.
2. To add a custom operation, click **New**.
   Or
3. To modify an existing custom operation, click its name in the table.
4. For the new custom operation, enter a name describing the new operation in the **Name** field.

For example, if the purpose of this operation is to extract information using a certain protocol, you can name it NetApp protocol query.

5. Enter or modify the operation purpose or description in the **Description** field.

6. Write the Java script in the **Script** pane to define the business logic of the operation.
   The script must comply to the following guidelines:

   - Create variables for operation parameters using the dollar sign in front of the variable name, for example, $fileName.
   - CTX is an object containing all the information which resulted from a pattern execution.
   - Use rtrn to indicate the string, which is the result of your custom operation.

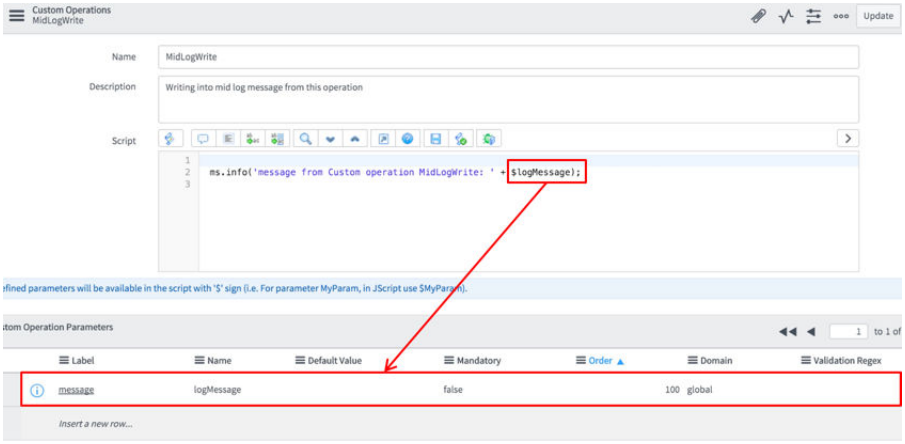
   For example:

```
ms.info('message from Custom operation MIDLogWrite '
+ $logMessage);
```

   Where ms.info is the MID Server log. The $logMessage variable refers to the message that is created inside this MID Server log.

7. In the **Custom Operation Parameters** related list, define the input parameters you used in the Java script.

   For example, if you used a variable for the log message file in the Java script, you must also define the parameter for this variable:



Field	Description
Label	Enter a short and descriptive label for the parameter. The label shows in the table. For example, netapp_query.
Name	Enter a short and clear name for the parameter without spaces, for example, NetAppquery. This name appears in the list of operations on the pattern step page.
Default value	If relevant, enter the default value. For example, POST.
Mandatory	Set to true if this parameter is mandatory for this operation. Otherwise, set to false.
Order	Define the order in which parameters appear on the operation page. Use natural numbers, where 1 means this

Field	Description
	parameter appears at the top of the operation page.
Domain	For domain-separated environments, specify the name of the domain for which this parameter is relevant.
Validation Regex	Specify a Java regular expression to validate the parameter with. During discovery process, the MID Server validates the operation result. The Pattern Designer module also runs validation when you click **Test** on the pattern step page.

Related topics

- Syntax editor

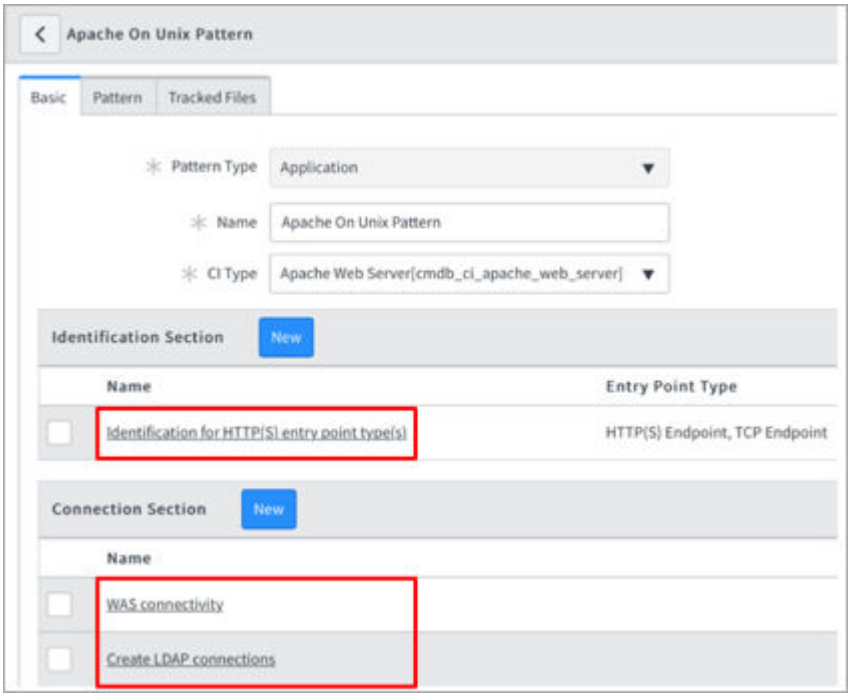
Explore remote host

Check details of file structure, setup, or parameters and values inside specific files on a host before using this information in step operations.

Before you begin

Basic knowledge of programming is desirable.

1. Navigate to **Pattern Designer > Discovery Patterns**.
2. Select a pattern from the Discovery Patterns list.
3. Select an identification or connection section.



4. Switch to the debug mode as described in Activate pattern Debug mode.

Role required: pd_admin

About this task

Part of creating or modifying patterns is defining step operations that use information about the host on which they run. You can verify that the information you are going to use in operations is correct, by exploring hosts.

You can explore both remote hosts and MID Servers.

Procedure

1. Click **Command Prompt** next to the Debug Mode button.
   The Command Prompt window opens.
2. Enter the command.

To make the platform apply applicative credentials while executing this command, use placeholders for credentials using the following syntax:

- $$username$$ - for the user name
- $$password$$ - for the password

For example, a parsing command for a Microsoft SQL Server uses credential placeholders:

```
"su - " + $userid + " -c '" + $ExecutableDir +
"mysql --user=" + "$$username$$" + " --password=" + "$
$password$$"
```

3. (Optional) Click **Advanced Details**.
4. (Optional) Specify the command and parameters:

Field	Description
Execute Mode	Select the relevant option of running this command: • **Default (Remote)** - on a remote device • **Local Script** - on the MID Server • **Windows Service** - on a remote Windows server
CI Type	If necessary, select the CI type whose credentials the platform applies while running the command.
Host	Enter the management IP address of the host on which to run the command. You can also leave this value empty.

5. Click **Run Command**.

The result is displayed in the Output pane.

6. Click **Close** when finished checking the result.

Related topics

- Applicative credentials

Pattern variables

You use variables in discovery patterns to refer to parameters or attributes of the CI that the pattern discovers.

There are several kinds of variables used in discovery: global variables, CI attribute variables, and temporary variables.

Characteristics	Global variable	CI attributes	Temporary variables
Description	Refers to general parameters of a device or application: **computer system** Contains information about the host of the CI for which you create a pattern. **entry point** Contains information about the connection which serves	Refers to parameters defined for the main or related CI type.	Refers to a parameter used for a specific operation in a pattern step.

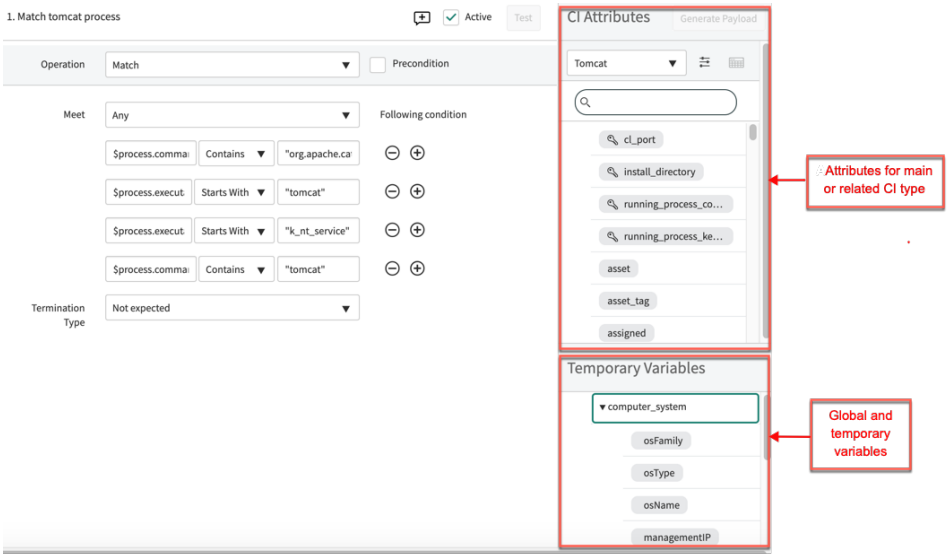
Characteristics	Global variable	CI attributes	Temporary variables
	as an entry point for the CI for which you create a pattern. **process** Contains data about the process detected at the port pointed to by the entry point. **Pre CMDB process** Contains data about the process detected on the IP and port of the entry point. Service Mapping retrieves this information from the CMDB before running patterns.		
Type/structure	Container variable: a variable can hold any combination of single strings	A variable can be: • Scalar: A single string • Tabular: A table, where	Scalar, tabular, or vector • Scalar: A single string • Tabular: A table, where

Characteristics	Global variable	CI attributes	Temporary variables
	and tabular variables.	each cell is a scalar variable. • Vector: A single, unnamed column with as many rows as needed	each cell is a scalar variable. • Vector: A single, unnamed column with as many rows as needed
Origin	Preconfigured in Service Mapping.	Service Mapping derives them from a CI type definition.	You create these variables while defining operations for pattern steps.
Modifiable in Pattern Designer	No	No	Yes

Some process variables retrieve values from the CMDB. Run the horizontal discovery prior to performing the top-down discovery to use these variables.

> **Note:** The Pre CMDB process variable is present for lightweight identification sections or for identification sections of patterns for discovery of load balancers.

Pattern Designer displays different kinds of variables in different areas of its interface.



Always prefix variables with the dollar symbol ($) which indicates variables, but is not actually a part of the variable name. For example, if you specify $Abc as the variable name, the actual name of the variable is Abc.

Change credentials to non-default

As part of creating or modifying a discovery pattern, you can define a **Change user** operation to make Service Mapping or Discovery use SSH or Windows credentials instead of the default administrative level credentials. This operation is relevant only for configuration items (CIs) hosted on Windows or Unix operating systems.

Before you begin

Role required: none

- Ensure that the OS credentials you want to use instead of the default credentials are correctly configured. Configure the credential alias to the OS CI type whose credentials you want to use. For operational information on how to create credentials, see SSH credentials and Windows credentials.
-

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

You can change the default credentials to any appropriate credentials belonging to a different CI type on the same or a different host.

Procedure

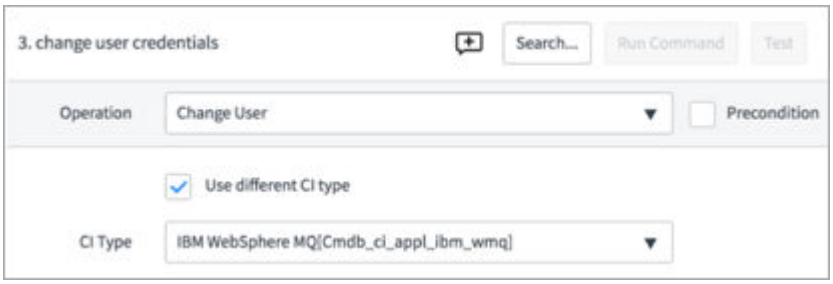
1. Select **Change user** from the **Operation** list in one of the following locations.
2. Select the **Use Different CI Type** check box.
3. From the list, select the CI type whose credentials you want to use.
4. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern

This operation is used in	This item
Section	Local queue connectivity
Step number and Name	3. Change user credentials

As part of discovery of an IBM WebSphere MQ Queue, you change credentials to IBM WebSphere MQ. It allows you to manipulate local IPs to discover connections on the local queue.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related tasks

- Change credentials to default

Provide connection information

As part of creating or modifying a discovery pattern, use the **Create connection** operation to provide information about an outgoing connection. This operation is only available for connection sections.

Before you begin

Role required: none

Navigate to the relevant step:

1. On the pattern form, select the relevant connection section.
2. Select the relevant step or create a pattern step by clicking  .

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

Always start with this step when creating a connection section entry for a new pattern.

Procedure

1. Select **Create Connection** from the **Operation** list.
2. Fill in the fields, as appropriate.

Field	Description
Select Connection Type	• **Application Flow**: Used between two applications (can be of the same type). • **Cluster**: Used for connections to CIs of the cluster type. Specify a cluster name. • **Inclusion**: Used for connections to an object that is included in the current object. For example, a connection from J2EE to EAR, and a connection from IIS to a website. • **Storage Flow**: Used for connections between configuration items (CIs) of

Field	Description
	host type and devices of storage type.
Select Entry Point	Select the entry point type from the list. For more information, see Entry point attributes.
Enter Connection Attributes	Define attributes by either entering actual values or variables.
Select Target CI Type	Select the target CI type from the list.
Is hidden	Select the check box if the connection should be hidden (that is, not shown in the user interface), but is used for continuing the discovery flow.
Is traffic based	Select the check box to use the traffic-based discovery method for this connection.

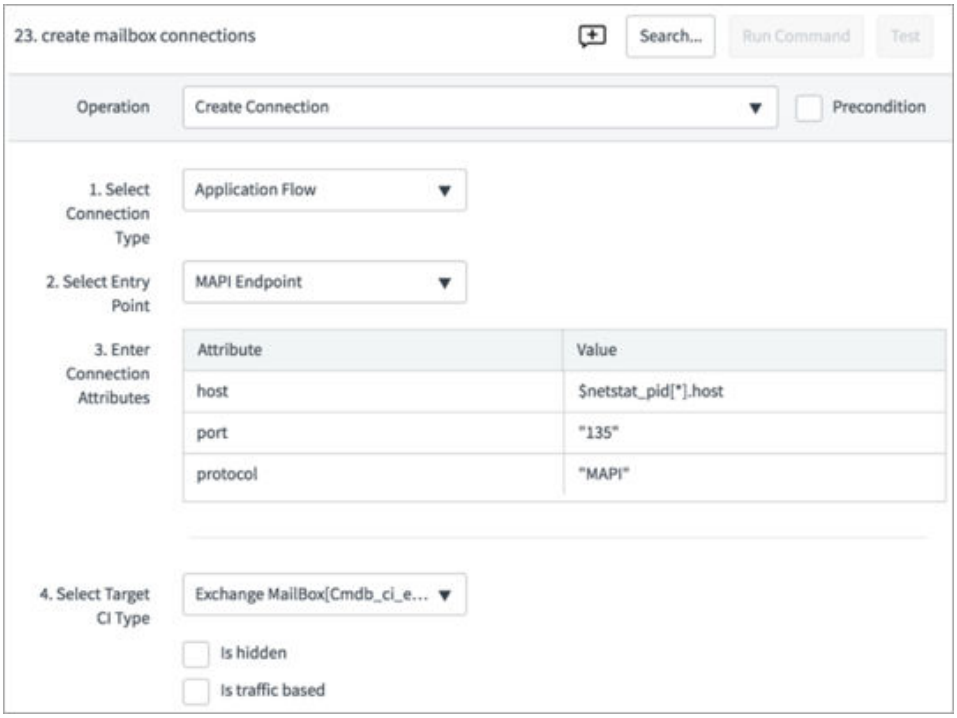
3. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern
Section	Alias queues connectivity

This operation is used in	This item
Step number and Name	6. Create outgoing connection to alias queues

To create a connection from Microsoft Exchange CAS to Exchange Mailboxes, define the **Create Connection** operation as follows:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Merge tables

As part of creating or modifying a discovery pattern, you can use the **Merge Table** operation to merge content from two source tables into a target table.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

Use this operation to unify information from different sources.

For example, during discovery of connections of IBM WebSphere Portal to IBM WebSphere MQ, you can merge the table containing queue names with the table containing JNDI reference names.

Procedure

1. Select **Merge Table** from the **Operation** list.
2. Fill in the fields, as appropriate.

Field	Description
First table	Specify the name of the first source table.

Field	Description
Second table	Specify the name of the second source table.
Target table	To create a table to contain merged data, define the name of the new table. To add information from one source table into another, specify the name of the source table to contain merged data.
Unmatched values	For tables that meet the merge criteria, select an action from the list for unmatched rows: • **Keep**: If merge criteria is met for any row, it merges all rows from both source tables into the target table. • **Remove**: Merges only matching rows from both source tables into the target table, and excludes non-matching rows.

3. To merge tables based on one matching field value:
    a. Click **Field matching**.
    b. In the **First Table Field**, enter the value from the first source table.
    c. In the **Second Table Field**, enter the value from the second source table.
4. To merge tables based on multiple matching field values:
    a. Click **Condition**.
    b. In the first condition field, enter the required value.

For example, enter the actual string or a variable name.

You can use variables including values from tabular variables as described in Enter values and variables in patterns.

c. Select an operator from the list.

d. To add more conditions, click the plus icon and define the criteria.

e. If you create multiple conditions, define if this pattern must meet all or any of them: From the **Meet** list, select `All` or `Any`.

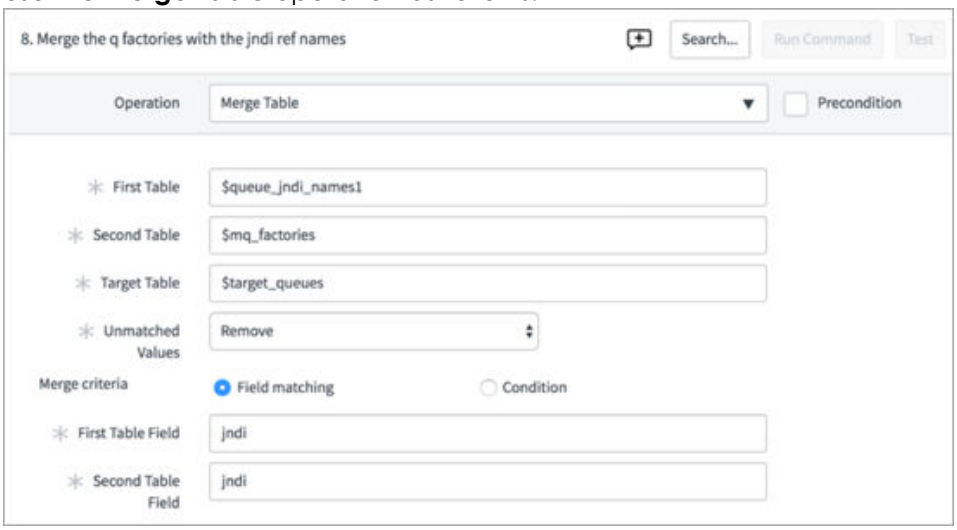
5. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Application Servers
CI Type	Websphere Portal [cmdb_ci_appl_websphere_portal]
Pattern	Websphere Portal On Linux
Section	EAR TO MQ Connectivity

This operation is used in	This item
Step number and Name	8. Merge the q factories with the jndi ref names

To merge tables containing queue names with JNDI reference names, use the **Merge Table** operation as follows:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Find a matching URL

As part of creating or modifying a discovery pattern, you can use the **Find Matching URL** operation to find the best match for the main URL and create a variable to hold it.

Before you begin

Role required: none

-

Make sure that you know the main URL of the application.

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

Some application CIs have configuration files containing lists of possible URLs. You can search such lists to find the match for the URL that allows connecting to a configuration item (CI).

Procedure

1. Select **Find matching URL** from the **Operation** list.
2. Fill in the fields, as appropriate.

Field	Description
Values	Specify the table and records containing potential URLs. You can use values from variables, including temporary tabular variables: from a specific field or a specific column in a table sequentially, starting from the first row. For more information, see Enter values and variables in patterns.

Field	Description
URL	Specify the URL for which you are seeking the best match.
Target Variable	Specify the variable name to contain the best match.

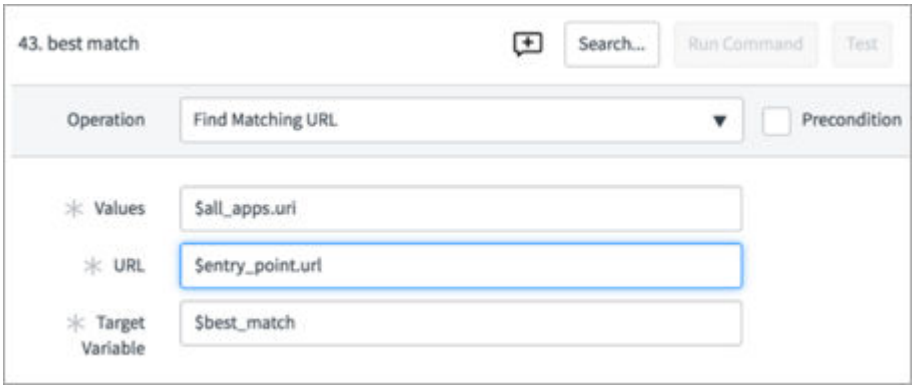
3. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Application Server
CI Type	Jboss [cmdb_ci_app_server_jboss]
Pattern	Jboss pattern
Section	http identification section
Step number and Name	43. best match

Example

To filter the table on Jboss and to create a variable to hold the best match for the Jboss entry point URL, define the **Find Matching URL** operation as follows:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Filter a table

As part of creating or modifying a discovery pattern, you can use the **Filter table** operation to search a source table for a specified value. If found, values are logged in a specified target table.

Before you begin

Role required: none

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Role required: pd_admin

Procedure

1. Select **Filter Table** from the **Operation** list.
2. Fill in the fields, as appropriate:

Field	Description
Source table	Name of source table to be searched. Use drag-and-drop or auto-complete to copy table names, fields, and values from the Temporary Variables and CI Attributes tables.
Target table	Name of target table to contain the values that were found. **Note:** If the specified target table is already populated, the new fields and values from the source table overwrite the existing fields and values of the target table.

3. Define the condition for this operation:
   a. In the first condition field, enter the required value.
   For example, the table name and the column from which to retrieve the values.

You can also use variables including values from tabular variables as described in Enter values and variables in patterns.

b. Select an operator from the list.

c. Enter the string that Pattern Designer uses to filter values.

d. To add more conditions, click the plus icon and define the criteria.

e. If you create multiple conditions, define if this pattern must meet all or any of them: From the **Meet** list, select `All` or `Any`.

4. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

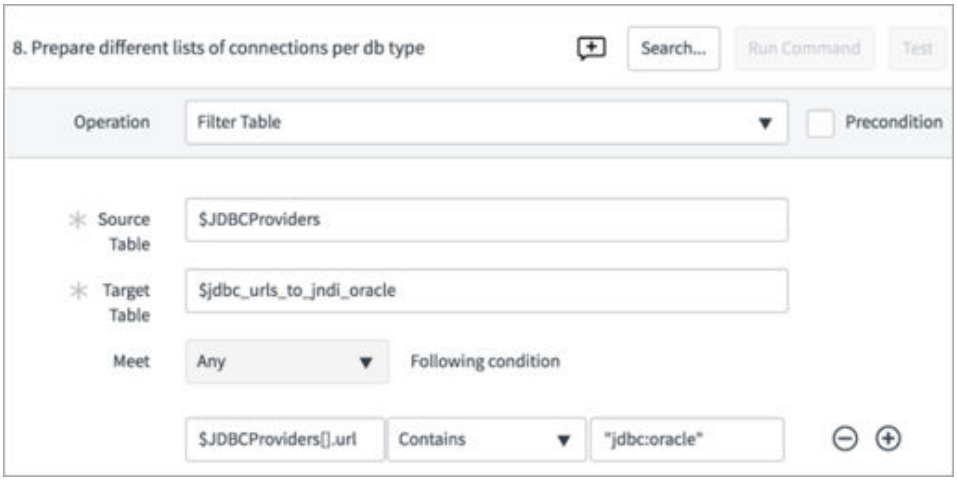
Example

This operation is used in	This item
Hierarchy	Applications > Application Servers
CI Type	WebSphere Portal [cmdb_ci_appl_websphere_portal]
Pattern	Websphere Portal On Linux
Section	DB2 JDBC connectivity

This operation is used in	This item
Step number and Name	8. Prepare different lists of connections per db type

Example

To identify IBM WebSphere connections to Oracle applications, use the **Filter Table** operation as follows:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Get a process

As part of creating or modifying a discovery pattern, use the **Get process** operation to search for a specific process to store in a tabular variable.

Before you begin

Role required: none

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

You can manually specify the filtering criteria, or you can select a process from the list of all processes on the system. The values of the selected process are used to populate the filtering fields. Modify these criteria as needed (for example, to delete irrelevant criteria).

Processes that satisfy the specified filtering criteria are placed in a tabular variable whose name you specify. This tabular variable appears in the **Temporary Variables** table.

Procedure

1. Select **Get process** from the **Operation** list.
2. If in the debug mode, perform the following steps:
   a. Click **Browse Process** to open a form containing a list of processes.
   b. Select a process and click **OK**.
   Filtering criteria are populated with values from the selected process.
3. If not working in the debug mode, define the field values as needed:

Field	Description
Process ID	Enter the process ID. Not recommended since this parameter can be modified.
Command Line	Enter the command line or a string which is part of it. For example, you can use "bw" to find "bwengine."
Working Directory	Enter the working directory for the process. Not recommended, since this parameter can vary on different hosts.
Parent Process	Enter the process which is the parent of the process that you want to extract. Not recommended since this parameter can be modified or can extract many irrelevant sub processes.
Port	Enter the port on which the process runs. Not all processes are based on ports.

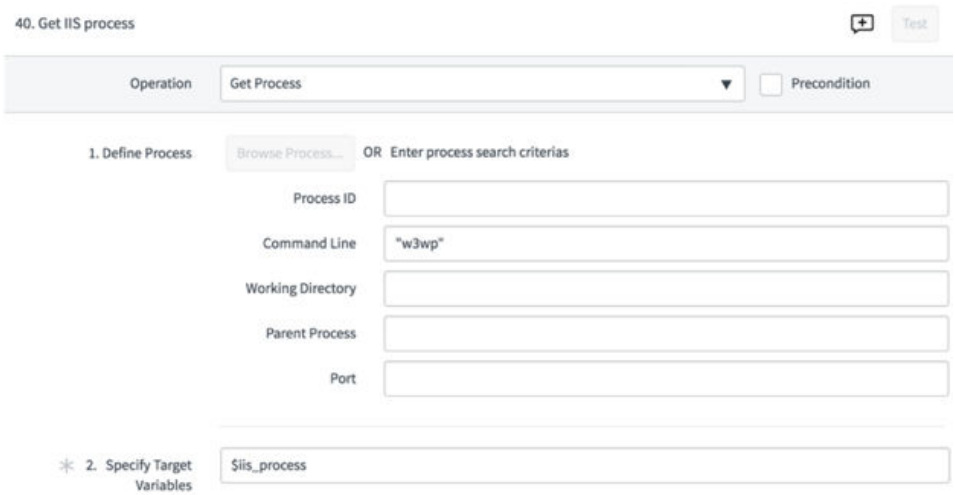
4. In the **Specify Target Variables** field, specify the name for the tabular variable to hold the list of processes that satisfy the filtering criteria. You can also enter a value from the specific field in a tabular variable as described in Enter values and variables in patterns.

5. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Application> Infrastructure Service > Web Server
CI Type	Microsoft iis Web Server [cmdb_ci_microsoft_iis_web_server]
Pattern	IIS
Section	Identification for HTTP(S) entry point type(s) for IIS6 second logic
Step number and Name	40. Get IIS process

Use the Get Process operation to get all Internet Information Services worker processes (w3wp.exe) running on the Windows Server and keep the results in the table variable iis_process.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Get a registry key

As part of creating or modifying a discovery pattern, you can use the **Get registry key** operation to retrieve and select registry key attributes to store in a table.

Before you begin

Role required: none

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

This operation is relevant only for Windows.

Procedure

1. Select **Get registry key** from the **Operation** list.
2. If working in the Debut Mode, perform the following steps:
   a. Click **Browse** and select the registry key.

The selected key path is placed in the **Registry key path** field. A form opens and displays a list of keys next to the tree.

You can use variables. You can also enter a value from the specific field in a tabular variable as described in Enter values and variables in patterns.

b. Select the key to show the attributes and click **OK**.

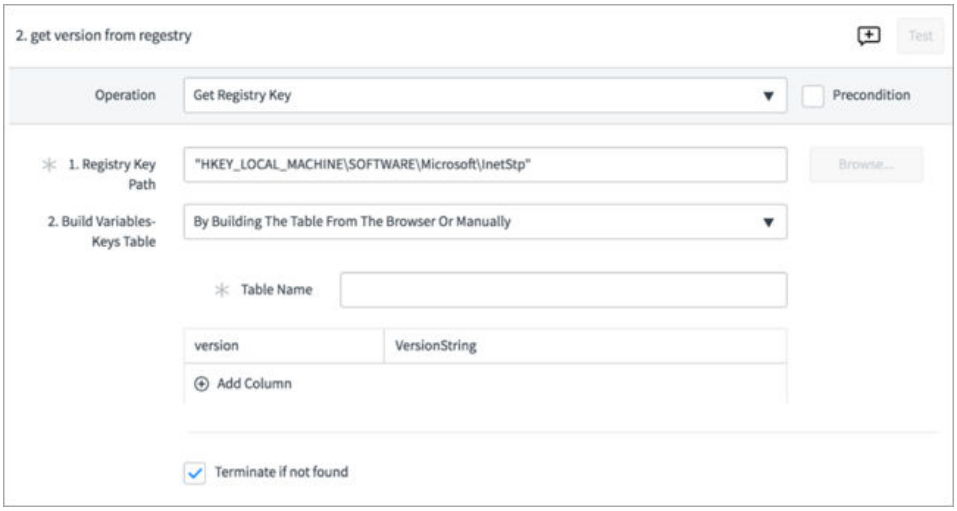
3. If not working in the Debug Mode, specify the registry key path in the **Registry key path** field.
4. Select the relevant option from the **Build Variables-Keys Table** list:
   - Select **By Using All Keys From The Registry Directory**, and enter the name of the table to contain the keys and variables.
   - Select **By Building The Table From The Browser or Manually**, and specify the keys on which you want to build the table. If there is more than one key, it creates a table to hold variables.
5. Define the name of the table to which you want to save the operation result.
6. Select **Terminate** to stop discovery if no results are found.
7. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

Field	Value
Hierarchy	Application > Infrastructure Server > Web Server
CI Type	Microsoft iis Web Server [cmdb_ci_microsoft_iis_web_server]
Pattern	IIS
Section	Identification for HTTP(s) entry point(s) for IIS6

Field	Value
Step number and Name	2. get version from registry

Get registry keys and values of "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\InetStp" and save the value in the "version" variable.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Match a condition

As part of creating or modifying a discovery pattern, you can use the **Match** operation to specify conditions that the discovery process must meet to continue. If these conditions are not met, the discovery process stops.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

If the specified conditions are not met, the discovery process always stops. At the same time, you can control if the system adds it to the discovery log as a discovery error or not. In some cases you may run a pattern containing the match operation and expect the result not to match conditions. For example, in cloud environments different datacenters may have different sets of devices and applications. You run a pattern for a certain application knowing that you do not find it on all datacenters.

If the lack of match is expected, the discovery log shows the message that you add instead of the discovery error. Customize the message to provide useful information about why this result is expected.

Procedure

1. Select **Match** from the **Operation** list.

2. In the first condition field, enter the required value.
   You can use an actual string or a variable. You can also use values from temporary tabular variables: from a specific field or a specific column in a table sequentially, starting from the first row. For more information, see Enter values and variables in patterns.

3. Select an operator from the list.
4. If necessary, enter the required value in the second condition field.
5. To add more conditions, click the plus icon and define the criteria.
6. If you create multiple conditions, define if this pattern must meet all or any of them: From the **Meet** list, select `All` or `Any`.
7. To consider the lack of match as expected:
   a. Select **Expected** from the **Termination type** field.
   b. Enter the text for the discovery message to provide useful information about the expected result.
   You can use variables in the discovery message. For example, you can enter the following text:

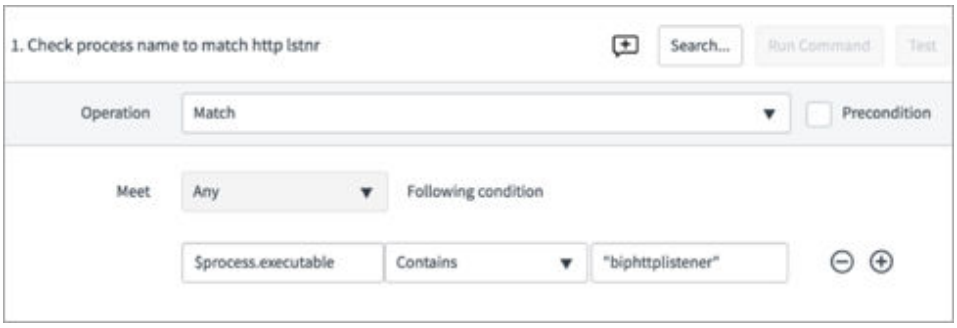
   ```
   The discovery was stopped due to the following error: $error.
   ```
8. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Business Integration Software

This operation is used in	This item
CI Type	IBM WMB HTTP Listener
Pattern	WMB HTTP Listener On Unix Pattern
Section	Identification for HTTP
Step number and Name	1. Check process name to match http lstnr

During discovery of an IBM WMB HTTP Listener, use the **Match** operation to check the process name.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Parse command output

As part of creating or modifying a discovery pattern, you can use the **Parse command output** operation to extract information from the command output and to save the operation result in a variable. You can also save the whole of command output as a variable.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. Select **Parse command output** from the **Operation** list.
2. Specify the command in **Set Command Details**.
   You can use variables. You can also enter a value from the specific field in a tabular variable as described in Enter values and variables in patterns.

   You can concatenate multiple commands.

   > **Important:** Avoid entering a specific path to a location or file because it can be different on different operating systems. You should use variables for paths.

   To make the Now Platform apply applicative credentials while executing this command, use placeholders for credentials using the following syntax:

   - $$username$$ - for the user name
   - $$password$$ - for the password

   For example, a parsing command for a Microsoft SQL Server uses credential placeholders: `"su - " + $userid + " -c '" + $ExecutableDir + "mysql --user=" + "$$username$$" + " --password=" + "$$password$$"`

3. To change the execution mode or credentials, click **Advanced** and fill in the fields, as appropriate.

Field	Description
Execute Mode	Select the relevant option for running this command: • **Default (Remote)** - on the remote device. • **Local Script** - on the MID Server • **Windows Service** - on the service running on the remote Windows server
CI Type	Enter the name of the CI type whose applicative credential you want to use for this step. The system uses applicative credentials for the defined CI type, different from the CI type, which this pattern discovers. For example, while discovering Microsoft SQL Server, you may switch to the Microsoft SQL instance applicative credentials. If there is more than one applicative credential for the selected CI type, the system uses applicative credentials according to their Order parameter. For more information about configuring applicative credentials, see Applicative credentials.

4. To save the whole of the command output as a variable, select `NONE` from the **Define Parsing** list and define the variable as described in 7.
5. Select the parsing strategy from the **Define Parsing** list.

Parsing strategy	Description
Oracle LDAP file XML file INI file Properties file JSON file (custom)	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more information, see Parse text using keyword, command, and positional type.

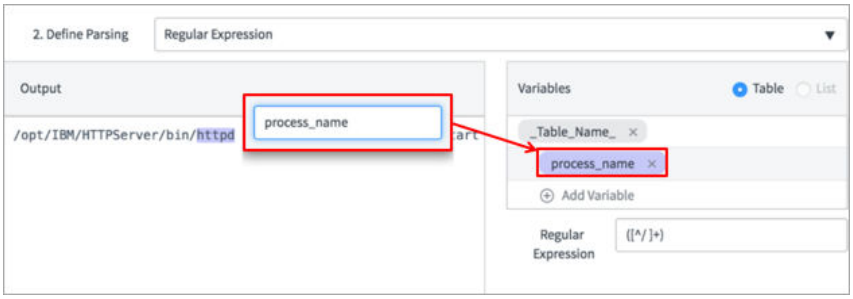
Parsing strategy	Description
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line. For more information, see Parse text using keyword, command, and positional type.
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic text files). See Parse text using delimited text.

6. If working in Debug mode, perform the following steps:

a. Click **Run Command** to see the result in the **Output** pane.

b. In the **Output** pane, mark text or symbols that you want a variable to contain.

c. In the variable name box, enter the name for the new variable, for example 'process_name'.

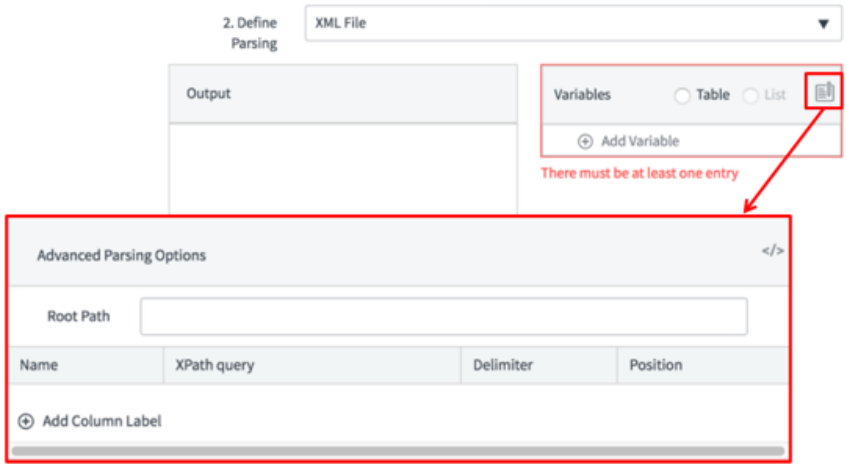


d. Press **Enter**.
The new variable is added in the **Variables** pane.

7. If you are not working in Debug Mode, perform the following steps to define the parsing criteria.

a. In the Variables pane, click **Add Variable** and enter the name for the new variable.

b. Click the **Advanced** icon.



c. Click **Add Column Label**.

d. Enter the parsing query.

e. Enter a value for the delimiter.

f. Enter a value for the position.

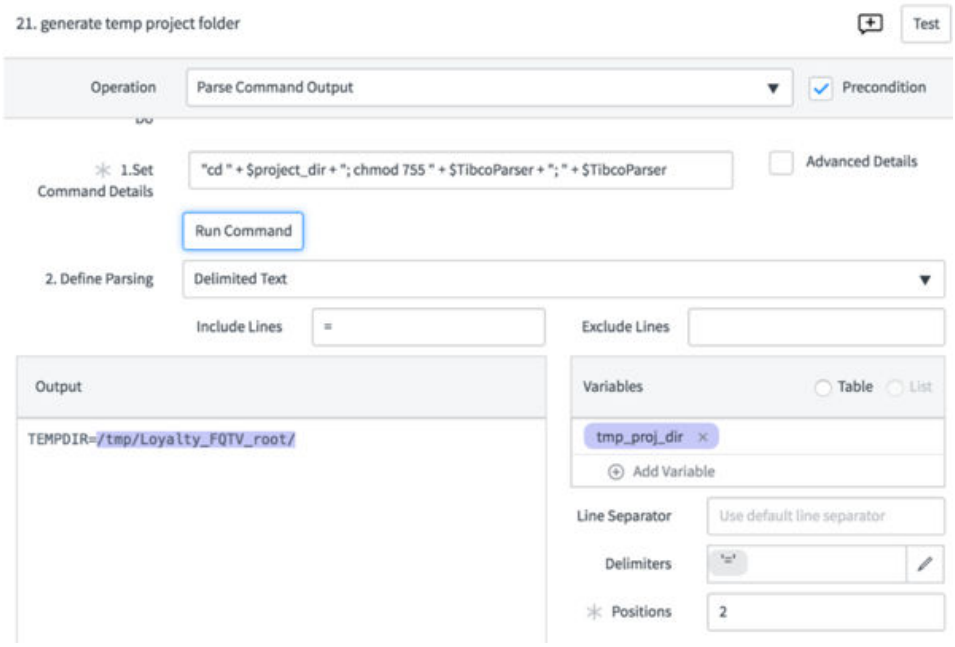
8. Select **Use Cache** to save the operation results in cache on the MID Server.
   Use cache to optimize discovery and avoid creating unnecessary load on central shared components, such as load balancers. The base system keeps operation results in cache for an hour.

9. Select **Terminate** to stop discovery if no results are found.

10. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

In this example, the Tibco parser script contained in the $TibcoParser variable runs on the target host. The parsing method is delimited text with the '=' delimiter. The result of the parsing is the path to the directory containing all configuration files for Tibco Business Works and Tibco Business Works Process. This step creates a temporary variable to hold the result of the parsing.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related tasks

- Activate pattern Debug mode
- Make a step conditional

Related reference

- Enter values and variables in patterns
- Pattern variables

Parse a file

As part of creating or modifying a discovery pattern, you can use the **Parse file** operation to extract information from a file and create variables to contain the extracted information.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

Alternatively, you can use traffic-based connections to create a parse file step in the relevant CI pattern as described in Fine-tune patterns using traffic-based discovery.

Procedure

1. Select **Parse file** from the **Operation** list.
2. Specify the file path in **Select file**.
   You can use variables. You can also use values from a temporary tabular variable: from a specific field or a specific column in a table sequentially, starting from the first row. For more information, see Enter values and variables in patterns.

   To specify the actual file, click **Browse**, navigate to the file, and click **Select**.

Important: Avoid entering a specific path to a location or file because it can be different on different operating systems. You should use variables for paths.

3. Select the relevant parsing strategy from the **Define Parsing** list and define parsing criteria.

Parsing strategy	Description
Oracle LDAP file XML file INI file Properties file JSON file (custom)	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more information, see Parse text

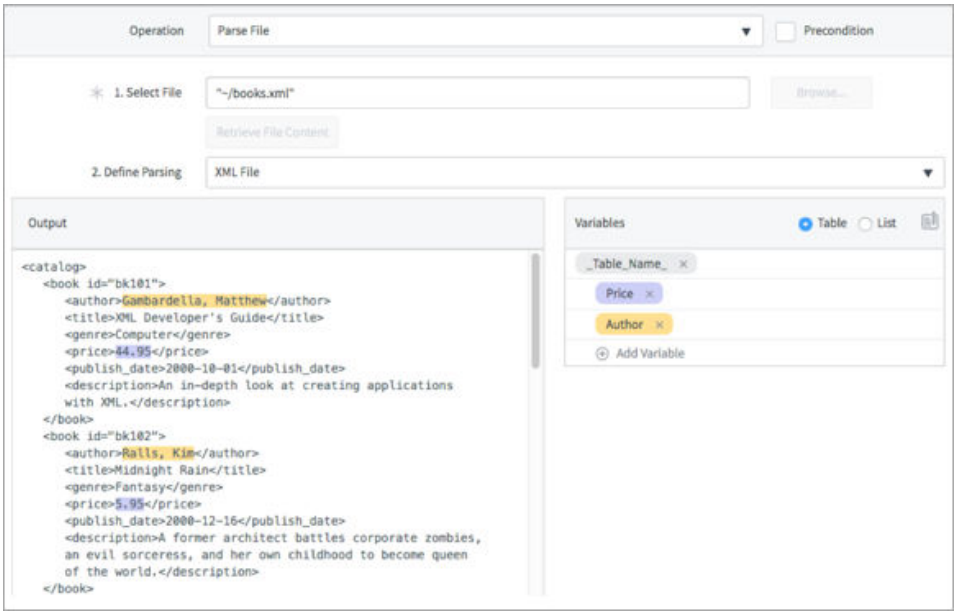
Parsing strategy	Description
	using keyword, command, and positional type.
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line. For more information, see Parse text using keyword, command, and positional type.
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic

Parsing strategy	Description
	text files). See Parse text using delimited text.

4. Select **Terminate** to stop discovery if no results are found.
5. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.
   In Debug mode, if any permission related errors are encountered during the operation, clicking on **Test** displays the related error message but the **Retrieve File Content** may still return the file content.

Example

You can use the Parse File operation to extract data on prices and authors from an xml file using the XML File parsing strategy.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.

- Finalize the pattern.

Parse a URL

As part of creating or modifying a discovery pattern, you can use the **Parse URL** operation to break down a URL to the component level.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

The URL parsing breaks a URL into the following components: protocol, hostname or IP, port, path and file.

Procedure

1. On the Identification or Connectivity Sections form, select **Parse URL** from the **Operation** list.
2. Fill in the fields, as appropriate.

Field	Description
Source	Specify the URL: • Enter a value that can change, a variable. Type the dollar character ($)

Field	Description
	and the first letters of the variable name. For example, `$ldap_url`. • Specify complex (concatenated) values in fields. Enter a value, then add a plus sign (+), and then enter another value. For example, `$install_directory+"conf/httpd.conf"`. • Enter constant values, a string. For example, `"/opt/ibm/mqsi/7.0/bin/"`. • Enter a value from a tabular variable: from a specific field or a specific column in a table sequentially, starting from the first row. For more information, see Enter values and variables in patterns. **Important:** Avoid entering a specific path to a location or file because it can be different on different operating systems. You should use variables for paths.
Target	Specify the table to hold the results. Use variables, complex values or strings as described above.

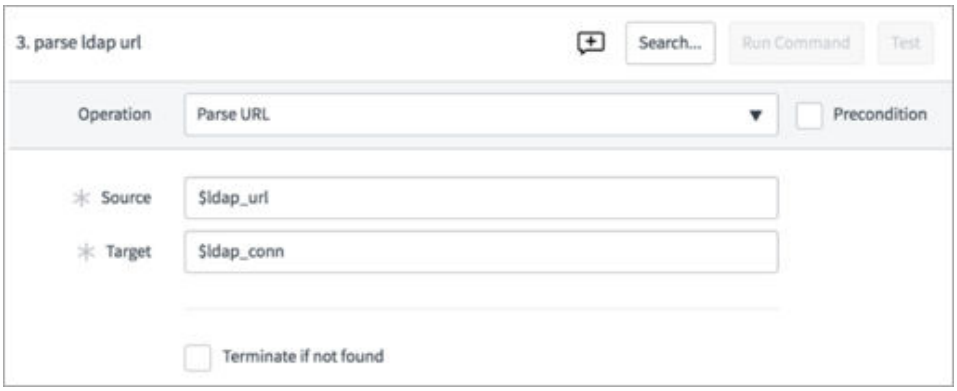
3. Select **Terminate** to stop discovery if no results are found.
4. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Application Servers

This operation is used in	This item
CI Type	Apache Web Server [cmdb_ci_apache_web_server]
Pattern	Apache On Unix Pattern
Section	Create LDAP connections
Step number and Name	3. parse ldap url

To break an LDAP directory URL to components and save the results in a new table, use the Parse URL operation as follows:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Parse a variable

As part of creating or modifying a discovery pattern, you can use the **Parse variable** operation to extract information from a variable and to store it in a variable table.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. Select **Parse variable** from the **Operation** list.
2. Define the variable that you wish to parse in the **Enter Variable** field. You can use regular variables. You can also use values from a temporary tabular variable: from a specific field or a specific column in a table sequentially, starting from the first row. For more information, see Enter values and variables in patterns.
3. Select the relevant parsing strategy from the **Define Parsing** list and define the parsing criteria.

Parsing strategy	Description
Oracle LDAP file XML file INI file Properties file	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.

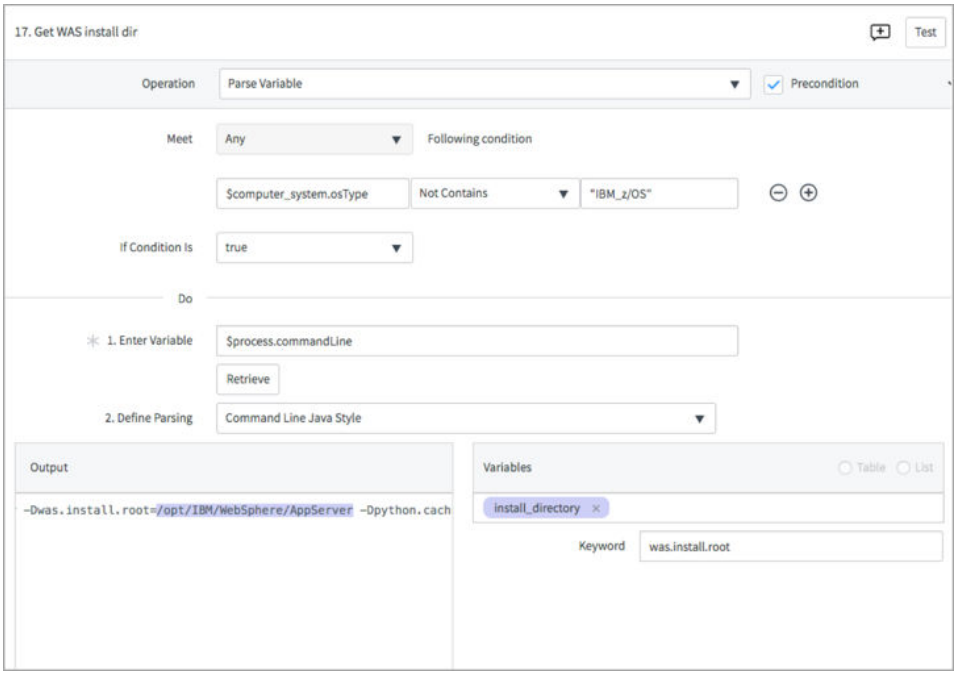
Parsing strategy	Description
JSON file (custom)	
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more information, see Parse text using keyword, command, and positional type.
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line. For more information, see Parse

Parsing strategy	Description
	text using keyword, command, and positional type.
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic text files). See Parse text using delimited text.

4. Select **Terminate** to stop discovery if no results are found.
5. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

You can use the Command Line Java Style parsing strategy to extract the path of the installation directory of the WebSphere Server.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Define an HTTP Get Call query

As part of creating or modifying a discovery pattern, you can use the **Http Get Call** operation to extract information from configuration items (CIs), which use the HTTP protocol.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. Select **Http Get Call** from the **Operation** list.
2. Define query parameters as necessary:

   Warning: If you customized this operation as described in Customize pattern operations, the query parameters may be different.

Field	Description
Required Authentication	If authentication is required for discovering the CI: • Set this parameter to **true**. • Configure credentials for the CI you want to discover as described in Basic authentication credentials. Set this parameter to **false** or leave empty if no authentication is required for accessing the CI.
URL	Specify the URL of the CI to discover using this pattern. You can use variables. You can also enter a value from the specific field in a tabular

Field	Description
	variable as described in Enter values and variables in patterns.

Headers	(Optional) Define HTTP headers to define the output format. For example, to see the output in the JSON format and bring in the security policy data, configure the header as follows:
	
	`content-type:text/JSON,`
	
	`content-security-policy:object-src 'none';base-uri 'self';script-src 'unsafe-inline' https: http: 'unsafe-eval';report-uri /_/NotificationsOgbUi/cspreport`

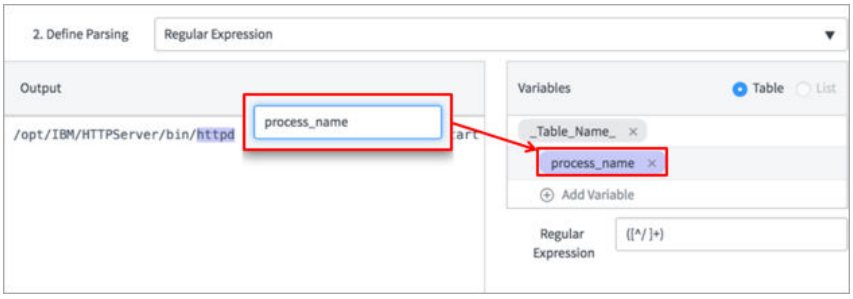
3. To save the command output in its entirety as a variable, select `NONE` from the **Define Parsing** list and define the variable as described in step 6.
4. Select the parsing strategy from the **Define Parsing** list.

Parsing strategy	Description
Oracle LDAP file XML file INI file Properties file	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.

Parsing strategy	Description
JSON file (custom)	
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more information, see Parse text using keyword, command, and positional type.
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line. For more information, see Parse

Parsing strategy	Description
	text using keyword, command, and positional type.
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic text files). See Parse text using delimited text.

5. If working in the Debug Mode, define the parsing criteria as follows:
   a. Click **Run Operation** to see the result in the **Output** pane.
   b. In the **Output** pane, mark text or symbols that you want a variable to contain.
   c. In the variable name box, enter the name for the new variable, for example 'process_name'.

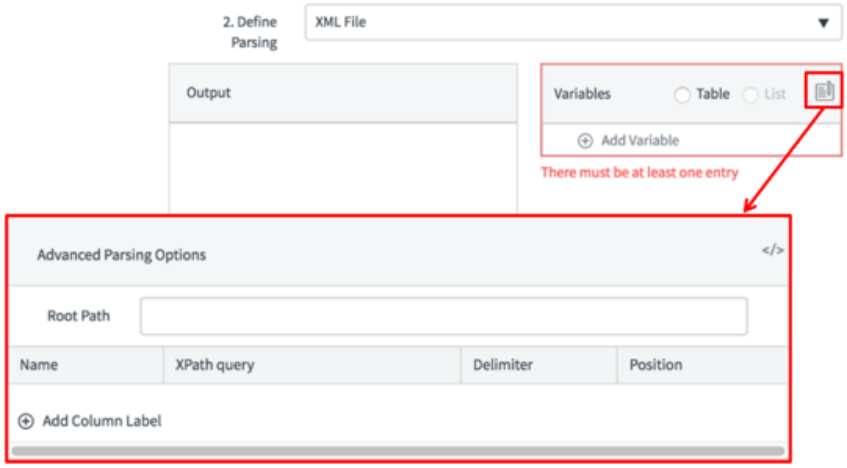


d. Press **Enter**.
The new variable is added in the **Variables** pane.

6. If you are not working in Debug Mode, perform the following steps to define the parsing criteria.

   a. In the Variables pane, click **Add Variable** and enter the name for the new variable.

   b. Click the **Advanced** icon.



   c. Click **Add Column Label**.

   d. Enter the parsing query.

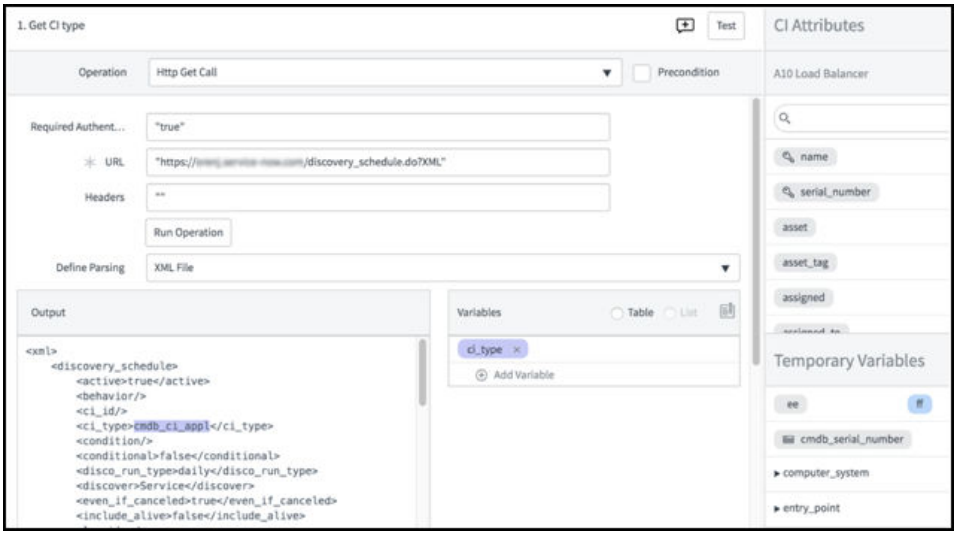
   e. Enter a value for the delimiter.

f. Enter a value for the position.

7. Select **Terminate** to stop discovery if no results are found.
8. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

You can use the HTTP Get Call operation to extract data on CI types using the XML File parsing strategy:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Define an API Query for cloud computing devices

As part of creating or modifying a discovery pattern, you can use the **Cloud REST Call** operation to extract information from configuration items of the PaaS (Platform as a Service) type, such as Microsoft Azure or Amazon Web Services.

Before you begin

Role required: pd_admin

- Verify that the operating system of the configuration item (CI) for which you want to use the Cloud REST Call operation extends the Logical Data Center OS type [cmdb_ci_logical_datacenter]:
  1. Navigate to the **Basic** tab of the CI pattern.
  2. Note the operating system for this CI.
  3. Navigate to **System Definition > Tables**.
  4. Set the search field to **Label** and enter the name of the operating system as stated on the **Basic** tab of the pattern.
  5. Find the operating system in the list and verify that Logical Datacenter appears in the **Extends table** column.
- Navigate to the relevant pattern step:
  1. On the pattern form, select the relevant identification section for Discovery.

     Alternatively, select the relevant identification or connection section for Service Mapping.
  2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

Deploy the Cloud REST Call operation in patterns used for discovery of PaaS CIs.

Refer to the official API-related documentation provided by manufacturers to obtain the query syntax for the device you want to query using the Cloud REST Call operation. For example:

- http://docs.aws.amazon.com/AWSEC2/latest/APIReference/Welcome.html

- https://docs.microsoft.com/en-us/rest/api/apimanagement/

Procedure

1. Select **Cloud REST Call** from the **Operation** list.
2. Define query parameters as necessary:

   Warning: If you customized this operation as described in Customize pattern operations, the query parameters may be different.

Field	Description
URL	Specify the URL as described in the official Microsoft Azure or Amazon Web Services documentation. You can use variables. You can use variables. You can also enter a value from the specific field in a tabular variable as described in Enter values and variables in patterns.
Method	Enter the method value as a string using all capital letters, for example "GET". The supported methods are: • GET • POST • PUT Refer to the relevant API guide for information. If the API documentation does specifically

Field	Description
	mention the HTTP query method, use the GET method.
Body	(Optional) Enter a request body as a string. If the string contains quotation marks, use the backslash mark in front the quotation marks to indicate that the string does not end at the quotation marks.
Headers	(Optional) If the relevant API documentation states that HTTP headers must be sent, enter these headers in the following format: Header_name1:header_value1,header_name2:header_value2 For example, `Content-Type:xml`.

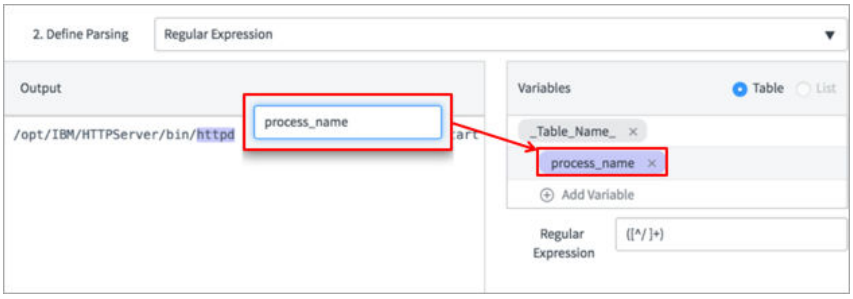
3. To save the command output in its entirety as a variable, select `NONE` from the **Define Parsing** list and define the variable as described in 6.
4. Select the parsing strategy from the **Define Parsing** list.

Parsing strategy	Description
Oracle LDAP file XML file	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.

Parsing strategy	Description
INI file Properties file JSON file (custom)	
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more information, see Parse text using keyword, command, and positional type.
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line.

Parsing strategy	Description
	For more information, see Parse text using keyword, command, and positional type.
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic text files). See Parse text using delimited text.

5. If working in the Debug Mode, define the parsing criteria as follows:
    a. Click **Run Operation** to see the result in the **Output** pane.
    b. In the **Output** pane, mark text or symbols that you want a variable to contain.
    c. In the variable name box, enter the name for the new variable, for example 'process_name'.

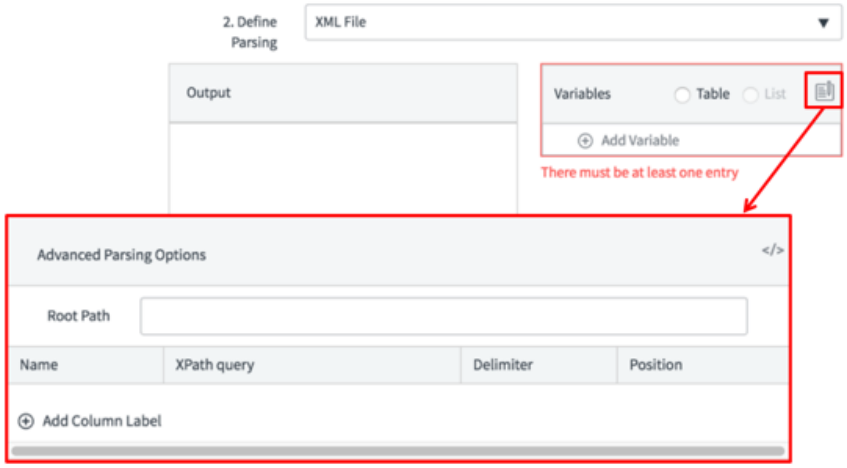


d. Press **Enter**.
The new variable is added in the **Variables** pane.

6. If you are not working in Debug Mode, perform the following steps to define the parsing criteria.

   a. In the Variables pane, click **Add Variable** and enter the name for the new variable.

   b. Click the **Advanced** icon.



   c. Click **Add Column Label**.

   d. Enter the parsing query.

   e. Enter a value for the delimiter.

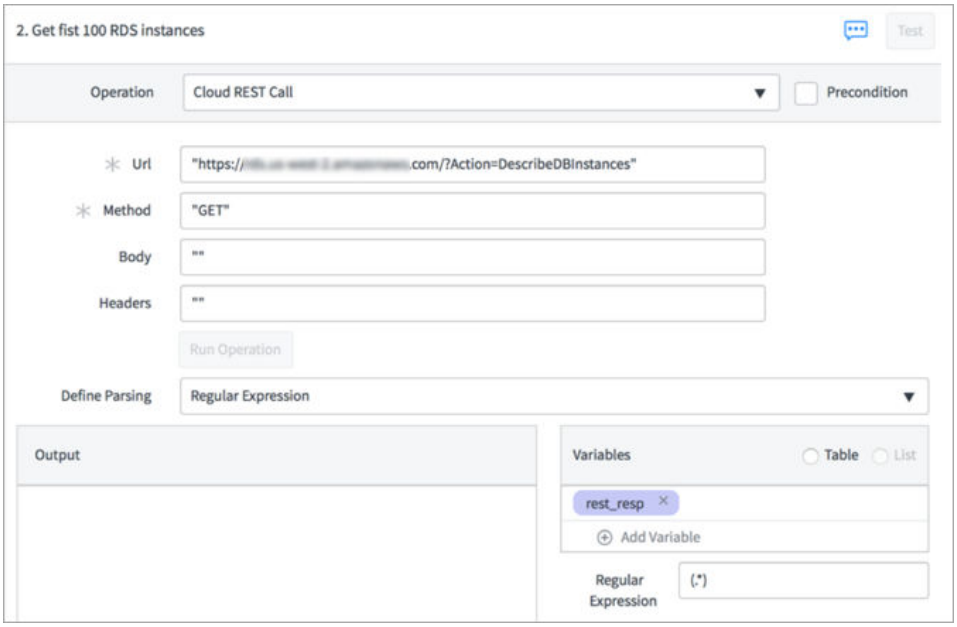
f. Enter a value for the position.

7. Select **Terminate** to stop discovery if no results are found.
8. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Application
CI Type	Load Balancer Service [cmdb_ci_lb_service]
Pattern	Amazon AWS Elastic Load Balancer Service
Section	AWS Load Balancer Service
Step number and Name	2. Get first 100 RDS instances

Use the Cloud REST Call operation to extract information on Database instances running on the region us-west-2 in the Amazon cloud.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related tasks

- Activate pattern Debug mode
- Make a step conditional

Related reference

- Enter values and variables in patterns
- Pattern variables

Create a relationship and a reference

As part of creating or modifying a discovery pattern, you can use the Relation and/or Reference condition to create relationships and references between CIs and their related items. This operation is relevant for both infrastructure and application patterns that Discovery uses for horizontal discovery.

Before you begin

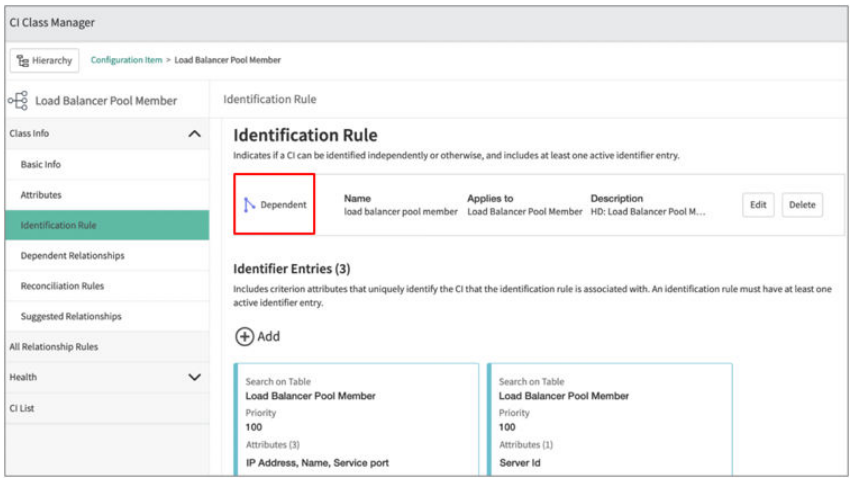
Role required: pd_admin

Decide on the nature of relationship or reference that you are going to create.

The CI relationship you define in the pattern must comply with the model you have created. Find out the existing relationships of the CI type for which you want to create a relationship or a reference:

1. Check if the CI type is dependent by performing the following steps:
    a. Navigate to CI Class Manager and click **Open Hierarchy**.
    b. Find the CI type in the hierarchy.
    c. Click **Identification Rule** on the left of the CI Class form.
    d. Check if the CI type is dependent or independent.

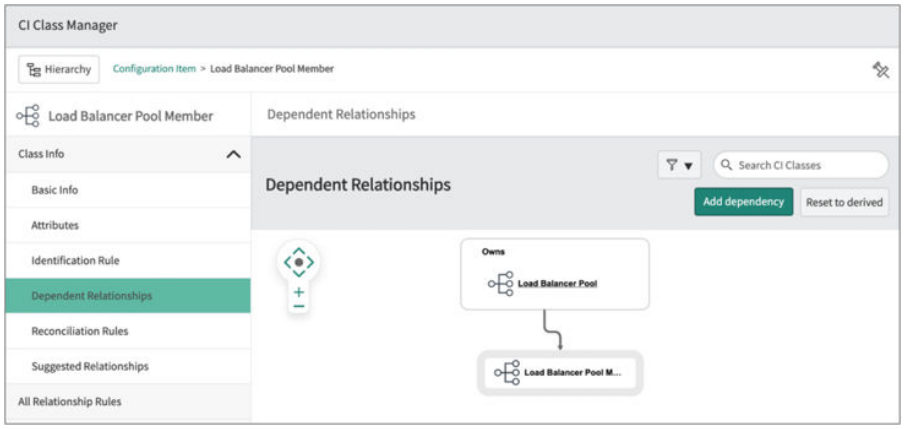
Example of a dependent rule for Load Balancer Pool Member



2. If the CI is dependent, click **Dependent Relationships** and check the dependent relationship rules.

   These rules define dependency structure of the CI types and the relationship types. For example, you can discover a load balancer pool with pool members belonging to it. To correctly reflect the type of relationship between these two CIs, create a containment rule, which defines the load balancer pool as the owner of pool members: **Owns::Owned by**. Refer to Dependent relationship rules to learn more.

   For independent CI types without dependent relationship rules defined for them, you can define any relationship.



3. (Optional for application patterns used for horizontal discovery) Correctly define operations preceding the Create Relationship/ Reference operation in the pattern. These operations must discover CIs for which you want to create a relationship or a reference. Use standard parsing operations to enter discovered CI data as temporary variables in the tabular format.

4. (Optional for creating a reference) Navigate to the relevant table definitions for the parent and the child CIs and choose the field to use as a reference field.

5. Navigate to the relevant pattern step:

   a. On the pattern form, select the relevant identification section.

   b. Select the relevant step or create a pattern step by clicking .

Basic knowledge of programming is desirable.

About this task

Discovery uses some patterns to discover a CI with all its related CIs and non-CIs: Items that do not extend the Configuration Item [cmdb_ci] table.

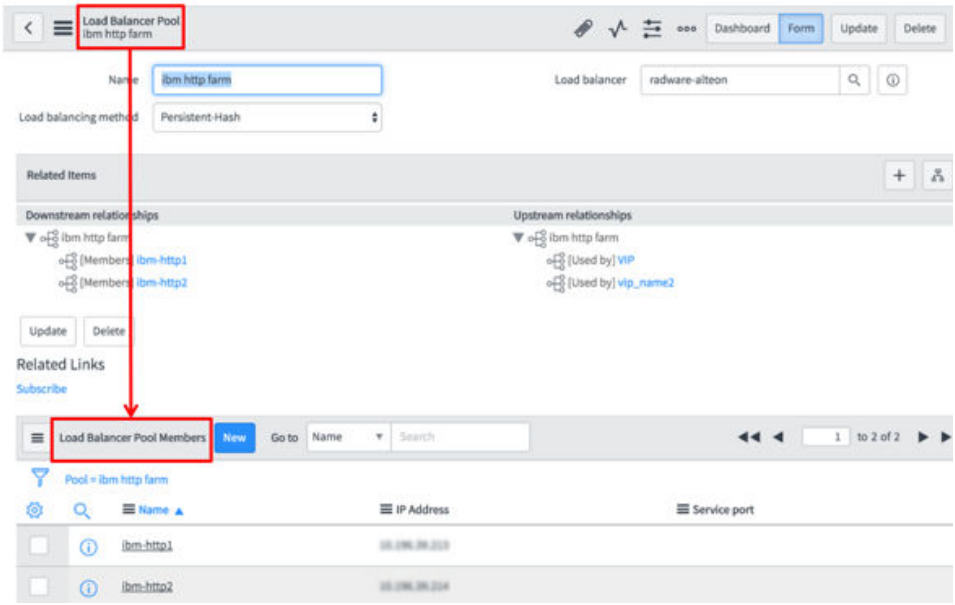
CIs can have different relationships. Dependent relationship rules describe relationships between CIs. The identification engine uses these

rules to identify CIs and determine if a specific CI exists in the CMDB or must be added to the CMDB.

If there is a relationship, the system uses the parent and child CI tables and creates a third table with data on the relationship between the parent and child CIs.

In addition to a relationship between CIs, you can create a reference connection between them. If a reference exists, then information about referenced child CIs appears at the bottom of the parent CI form. For example, all load balancer pool members appear on the form of the load balancer pool, which owns these members.

Referenced load balancer services appear on a load balancer form



Procedure

1. Discover related items together with the main CI to be able to view and use attributes of related CI types.
2. Select **Create Relation/Reference** from the **Operation** list.
3. Add the related CI type to this pattern as described in Discover related items together with the main CI.

4. Complete the form using the fields in the table.

Relation and/or Reference operation fields

Field	Description
Parent Table	Enter the name of the table for the CI that you want to use as a parent CI in the relation. For example, `cmdb_ci_lb_pool` for the load balancer pool CI.
Child Table	Enter the name of the table for the CI that you want to use as a child CI in the relation. For example, `cmdb_ci_lb_pool_member` for the load balancer pool member, which the load balancer pool owns.
Result Table	Enter the name for the new table to store information about relations and references between CIs, resulting from this operation.
Relation Type	Select the type of relationship between CIs from the specified target tables. The part in the type name before the separator (::) refers to the parent CI and the second part, after the separator (::), to the child CI. For example, in the **Owns::Owned by** option: • "Owns" indicates that a load balancer pool [cmdb_ci_lb_pool] owns a load balancer pool member [cmdb_ci_lb_pool_member].

Field	Description
	• "Owned by" indicates that a load balancer pool member [cmdb_ci_lb_pool_member] is owned by a load balancer pool [cmdb_ci_lb_pool]. **Warning:** Make sure to choose the right option among relation types using the same words, like **Used by::Uses** and **Uses::Used by**.
Reference	(Optional) Select this check box to create a reference between the parent CIs and the child CIs or non-CI related items.
Direction	(Only if **Reference** is selected) Select the direction of the reference: • **Parent to Child** to display the parent CI on the child CI form. For example, the load balancer pool member form displays load balancer pool, which owns this pool member. • **Child to Parent** to display child CIs on the parent CI form. For example, the load balancer pool form displays load balancer pool members this pool owns.
Column Name	(Only if **Reference** is selected) Enter the name of the field in the child CI table that the system uses to create the reference. For example, load_balancer.

Field	Description
Creation Criteria	Select the criteria used to create the relation and/or reference: **All** The system creates a relationship or reference without any additional criteria. **Field matching** The system creates the relationship or reference only if the value in the specified field in a parent CI matches the value of the specified field in a child CI. **Condition** The system creates the relationship or reference only if the specified criteria is matched.
Parent table field	(Only if **Field matching** is selected) Enter the field from the parent table to compare. You can use variables including values from tabular variables as described in Enter values and variables in patterns.
Child table field	(Only if **Field matching** is selected) Enter the field from the child table to compare.
Unmatched values	Operation for values in the two tables that do not match. You

Field	Description
	can keep or remove unmatched values.

5. If you selected **Condition** from **Creation Criteria**, define this condition:

   a. In the first condition field, enter the required value.

   b. In the first condition field, enter the required value.
   For example, enter the actual string or a variable name.

   c. Select an operator from the list.
   If you select `Is Empty`, the second field disappears.

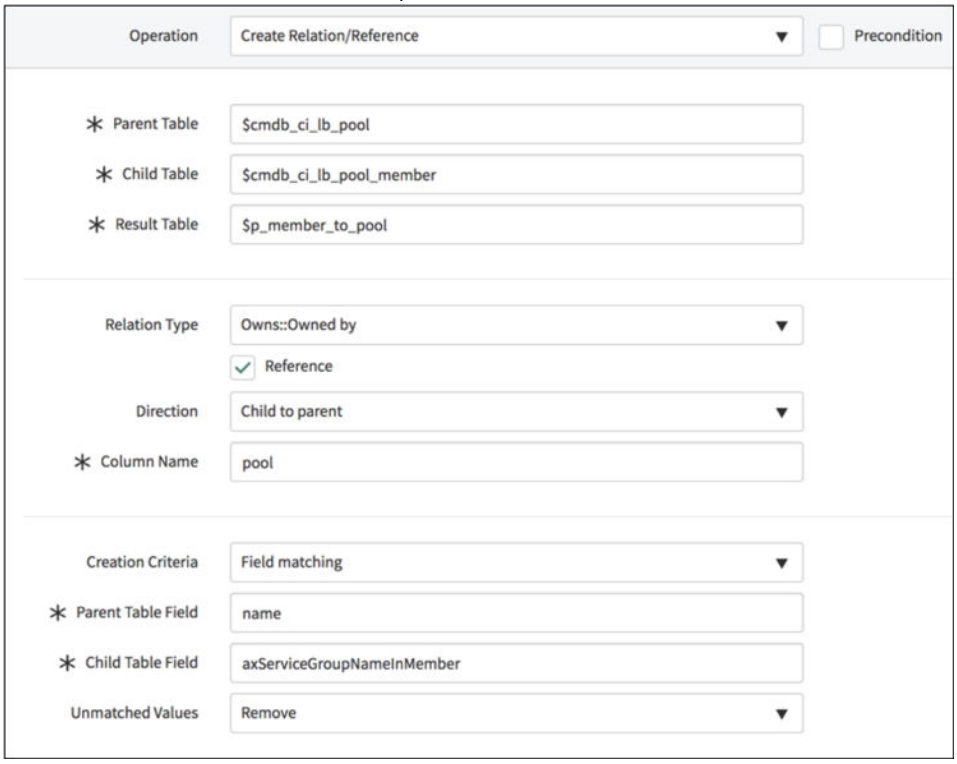
   d. To add more conditions, click the plus icon and define the criteria.

   e. If you create multiple conditions, define if this pattern must meet all or any of them: From the **Meet** list, select `All` or `Any`.

Example

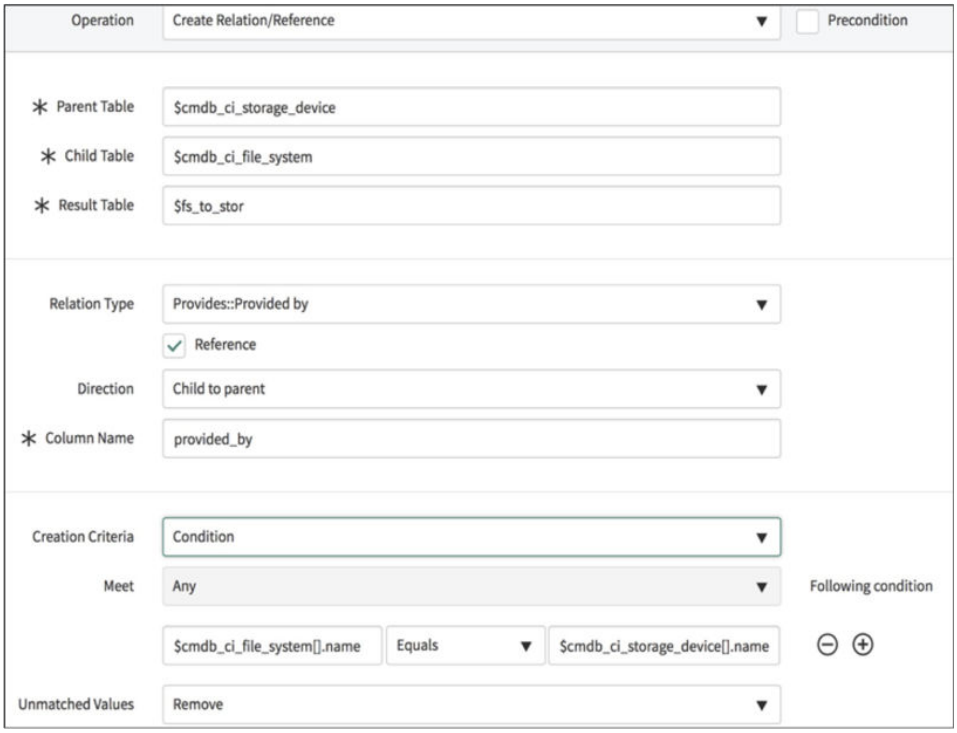
Use the **Create Relation/Reference** operation to create the **Owns::Owned by** relation between a load balancer pool and a load balancer pool member. You also create a reference from a load balancer pool

member to the load balancer pool.

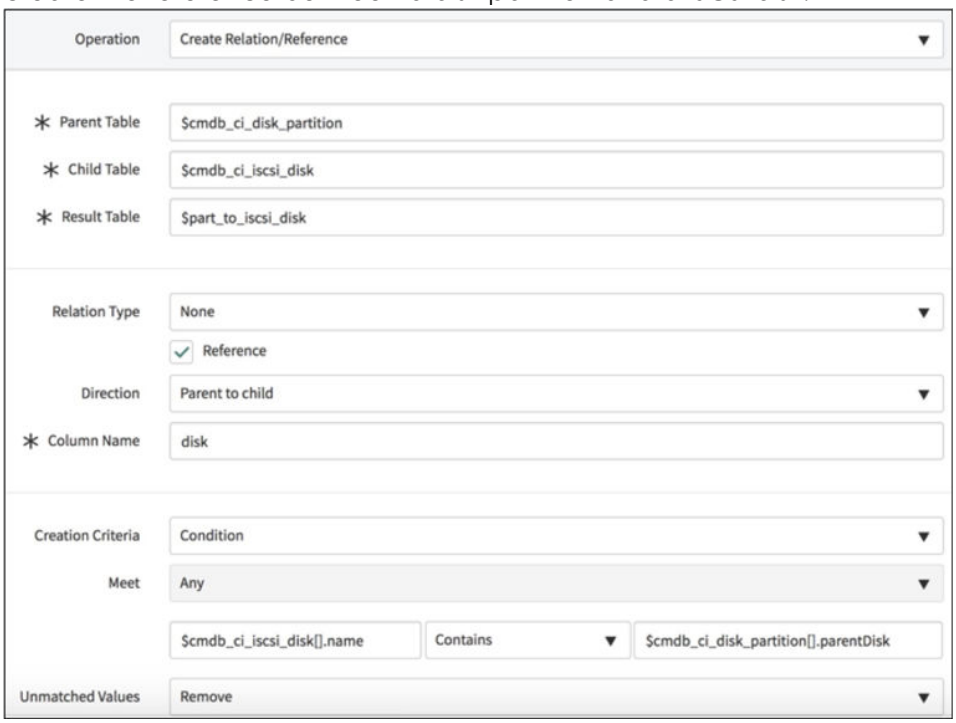


Another example is using the **Create Relation/Reference** operation to create the **Provides::Provided by** relation between a storage device and a file system. This operation also creates a reference from the storage

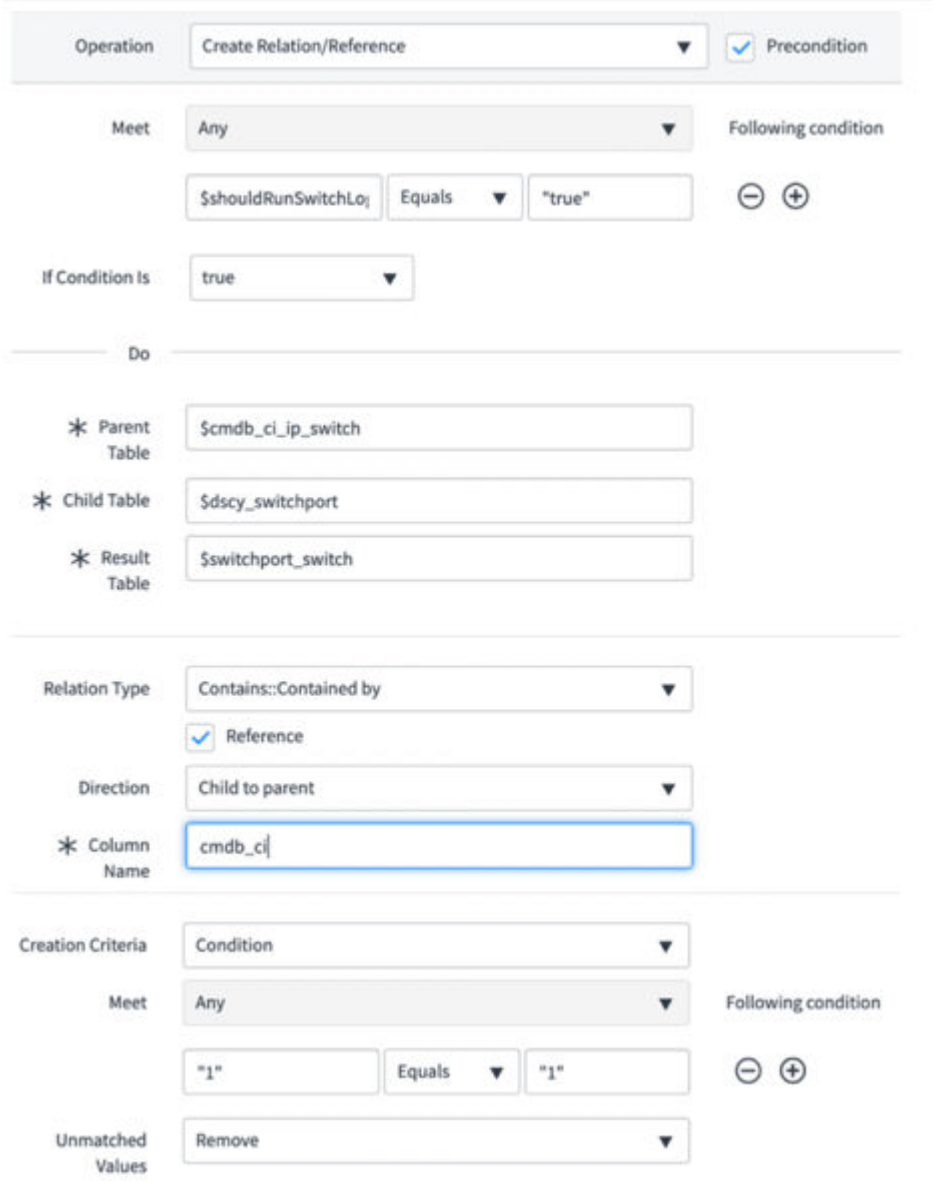
device to the load system.



Another example is using the **Create Relation/Reference** operation to create the reference between a disk partition and a iSCSI disk.

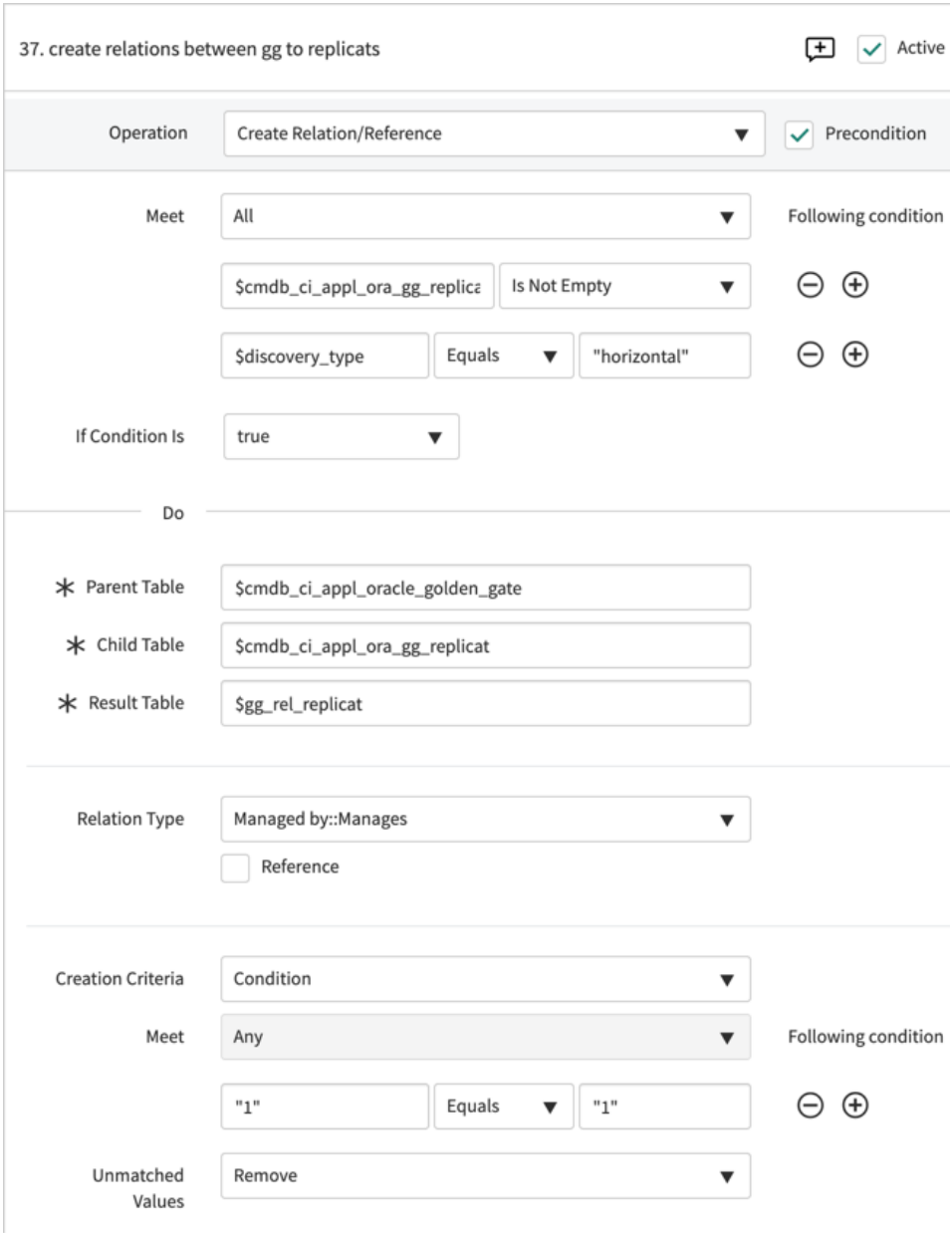


The following example shows how to add a non-CI item (a switch port [dscy_switchport]) to a CI (switch [cmdb_ci_ip_switch]).



You can also use the **Create Relation/Reference** operation to create the relation of the "Managed by::Manages" type between Oracle Golden Gate and Golden Gate Replica. In this case, you do not need to use the

Reference properties.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related topics

- Identification and reconciliation components and process

Reuse a shared step library

As part of creating or modifying a discovery pattern, you can reuse a sequence of discovery steps that you created for one pattern in other patterns. Saved shared step sequences become pattern modules referred to as shared libraries.

Before you begin

Role required: pd_admin

Basic knowledge of programming is desirable.

About this task

You do not have to recreate the same step manually if you reuse existing step sequences.

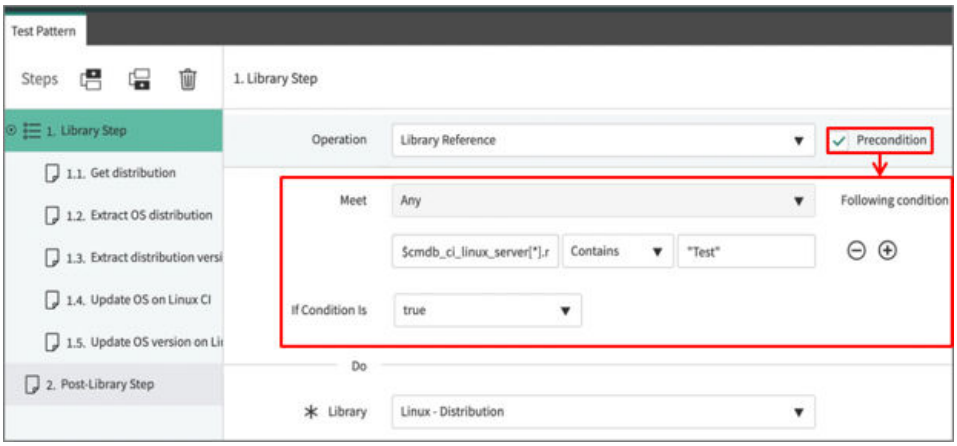
First, you save the steps you want to reuse into a repository as a shared library. Then you can insert them wherever relevant and as many times as needed.

You can deactivate shared libraries, so that patterns using them, skip steps in deactivated shared libraries. For more information, see Discovery Configuration Console. You can also disable the step containing a shared library for a specific pattern. This change does not affect other patterns containing the same shared library.

If a precondition defined for a shared library step is not fulfilled, the pattern skips all steps belonging to this shared library. For example, if the file, to which the variable points, does not contain "test", the pattern does not run the shared library steps 1-1 to 1-5. The precondition exists only for

the pattern within which you created it. The precondition does not affect other patterns using this shared library.

Library Reference operation form



Procedure

1. If the steps you want to reuse already exist, create a reusable step sequence as follows:
   a. Navigate to the pattern containing the steps that you want to reuse.
   b. Navigate to the steps that you want to combine.
   c. Select the steps, right-click, and choose **Create shared library**.
   d. Fill in the fields as appropriate:

Field	Description
Shared Library Name	Specify a unique name for the library. For example, UNIX-OS for a step sequence used to discover the UNIX operating system.

Field	Description
Library Description	Specify a meaningful description for the library.

e. Click **Create**.
The system saves the selected steps as a shared library in the Discovery Patterns [sa_pattern] table.

2. Reuse the step sequence you created:

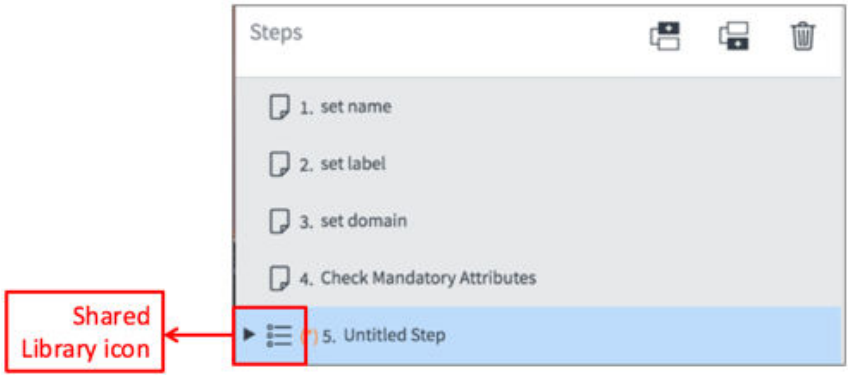
a. Navigate to the pattern in which you want to insert your step sequence.

b. Select the relevant identification section or connectivity section.

c. Select **Library Reference** from the **Operation** list in one of the following locations:

- the **Identification Sections** or **Connectivity Sections** for Service Mapping
- the Step window for Discovery

d. Select the required shared library from the **Library** list.
The library is inserted as substeps inside an untitled step.



e. Rename **Untitled Step** to reflect the purpose of the step.

3. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

4. To exclude the shared library from the pattern without deleting it, clear the **Active** check box.
   The MID Server skips the step containing the shared library while running the pattern.

What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Run an SSH script file

As part of creating or modifying a discovery pattern, you can use the **Run SSH Script File** operation to run composite commands or sequences of commands on Unix-based hosts.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. Select **Run SSH Script File** from the **Operation** list.
2. Enter file name of the SSH script file located on the MID Server in the **File Name**.

3. To save the command output in its entirety as a variable, select `NONE` from the **Define Parsing** list and define the variable as described in step 6.
4. If necessary, select the parsing strategy from the **Define Parsing** list.

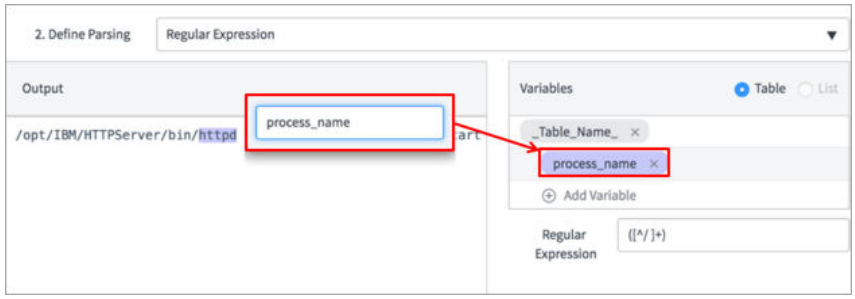
   Note: Alternatively, use EVAL scripts to manipulate collected data as described in KB0647736: Examples of EVAL scripts used in Discovery patterns.

Parsing strategy	Description
Oracle LDAP file XML file INI file Properties file JSON file (custom)	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more

Parsing strategy	Description
	information, see Parse text using keyword, command, and positional type.
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line. For more information, see Parse text using keyword, command, and positional type.
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic

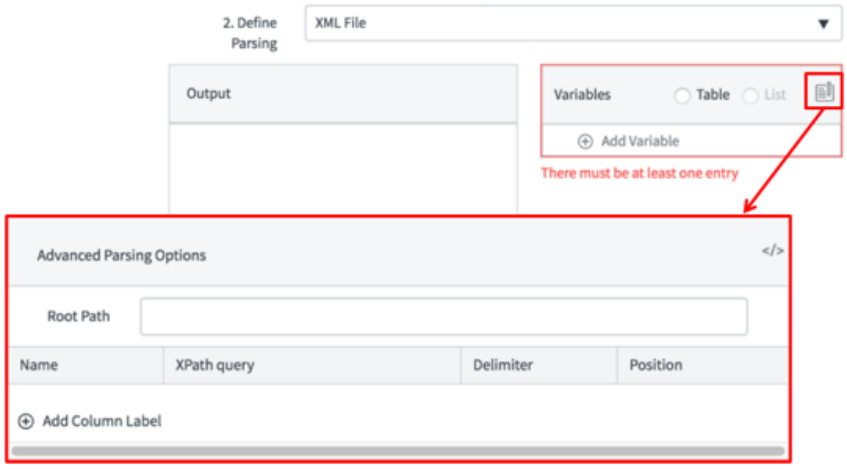
Parsing strategy	Description
	text files). See Parse text using delimited text.

5. If working in the Debug Mode, define the parsing criteria as follows:
   a. Click **Run Operation** to see the result in the **Output** pane.
   b. In the **Output** pane, mark text or symbols that you want a variable to contain.
   c. In the variable name box, enter the name for the new variable, for example 'process_name'.



   d. Press **Enter**.
   The new variable is added in the **Variables** pane.

6. If you are not working in Debug Mode, perform the following steps to define the parsing criteria.
   a. In the Variables pane, click **Add Variable** and enter the name for the new variable.
   b. Click the **Advanced** icon.



c. Click **Add Column Label**.

d. Enter the parsing query.

e. Enter a value for the delimiter.

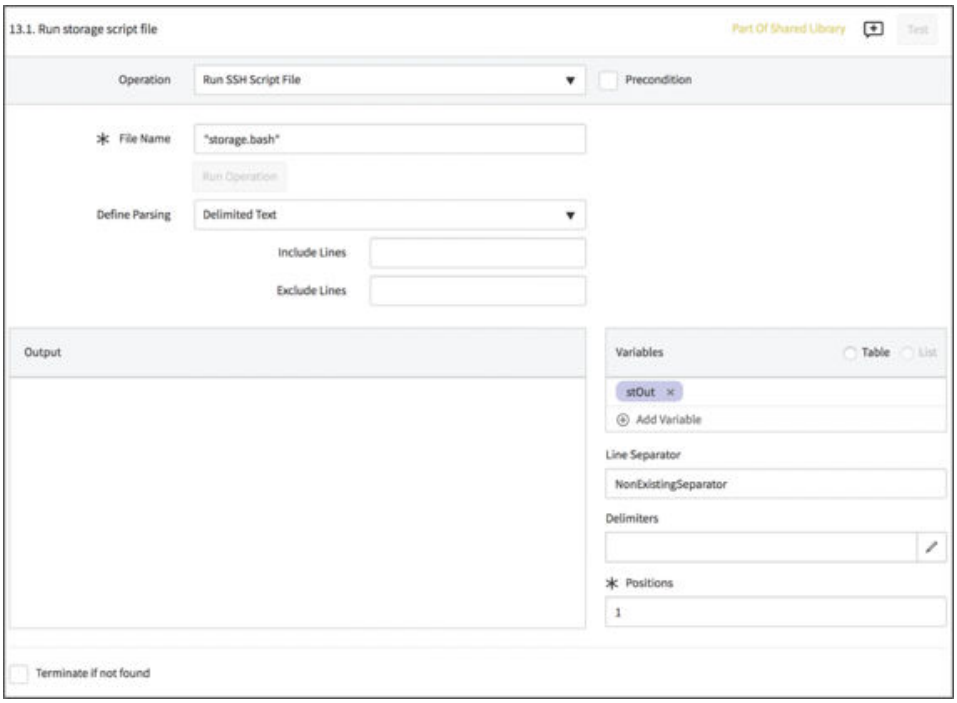
f. Enter a value for the position.

7. Select **Terminate** to stop discovery if no results are found.
8. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

Field	Value
Hierarchy	Hardware > Computer > Server > Linux Server
CI Type	Linux Server [cmdb_ci_linux_server]
Pattern	Linux Server
Section	discovery
Step number and Name	13.1 Run storage script file

This pattern step uses the Run SSH Script File operation to collect all information about storage entities associated with the discovered Linux server.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Set a parameter value

As part of creating or modifying a discovery pattern, you can use the **Set Parameter Value** operation to apply a value to a parameter.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. On the Identification or Connectivity Sections form, select **Set Parameter Value** from the **Operation** list.
2. Fill in the fields, as appropriate.

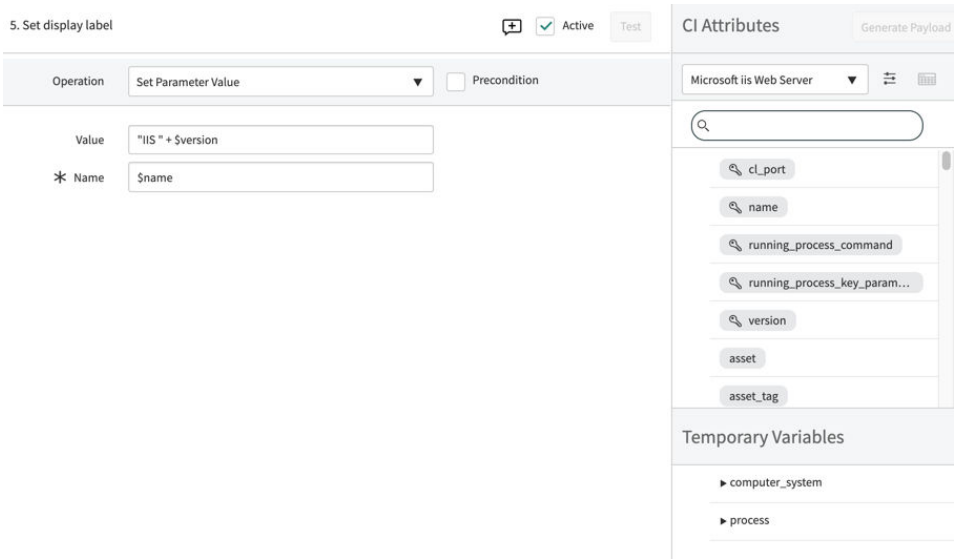
Field	Description
Value	Specify the value that you want to assign to the parameter. You can use strings or variables. You can also use values from temporary tabular variables: from a specific field or a specific column in a table sequentially, starting from the first row. For more information, see Enter values and variables in patterns.
Name	Specify the name of the variable to which to assign this value.

3. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

Field	Value
Hierarchy	Applications > Application
CI Type	Microsoft iis Web Server [cmdb_ci_microsoft_iis_web_server]
Pattern	IIS
Section	Identification for JHTTP(s) entry point.
Step number and Name	5. Set the display label

To set the CI name to "IIS" and the version number, define the **Set Parameter Value** operation as follows:



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Transfer a file

As part of creating or modifying a discovery pattern, you can transfer a file from the MID Server to a computer in your organization.

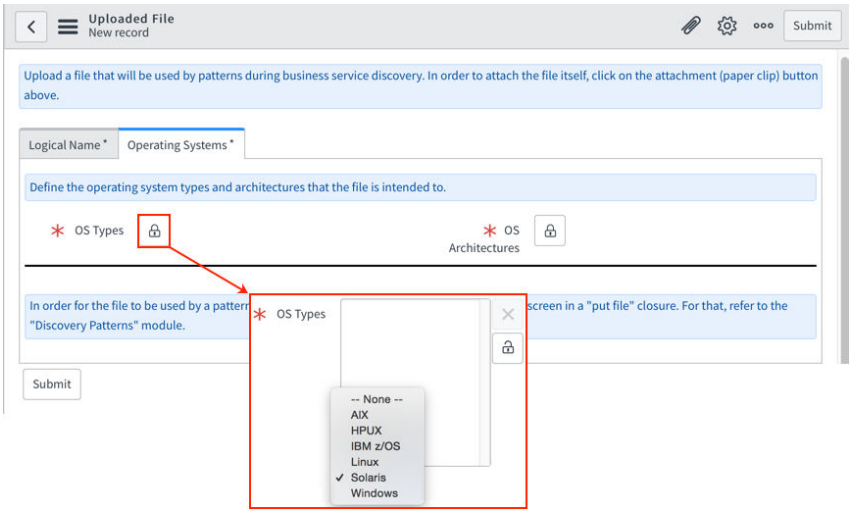
Before you begin

Role required: pd_admin

Basic knowledge of programming is desirable.

Procedure

1. Upload the file to the MID Server:
    a. Navigate to **Pattern Designer > Uploaded Files**.
    b. Click **New**.
    c. Enter a meaningful name in the **Name** field.
    d. Click the **Operating Systems** tab.
    e. Click **OS Types** and select the relevant type from the list.



f. Click **OS Architectures** and select the relevant option from the list.

> **Note:** You can select both 32-bit and 64-bit option for the same file if necessary.

g. Click the **Manage Attachments** icon.

h. Attach a file.

See Add and manage attachments.

2. Navigate to the relevant pattern step:

   a. On the pattern form, select the relevant identification section for Discovery.

      Alternatively, select the relevant identification or connection section for Service Mapping.

   b. Select the relevant pattern step or click  to add a step.

3. Select **Put file** from the **Operation** list.

4. Click **Refresh**.
   The name of the uploaded file appears in the **File name** field.

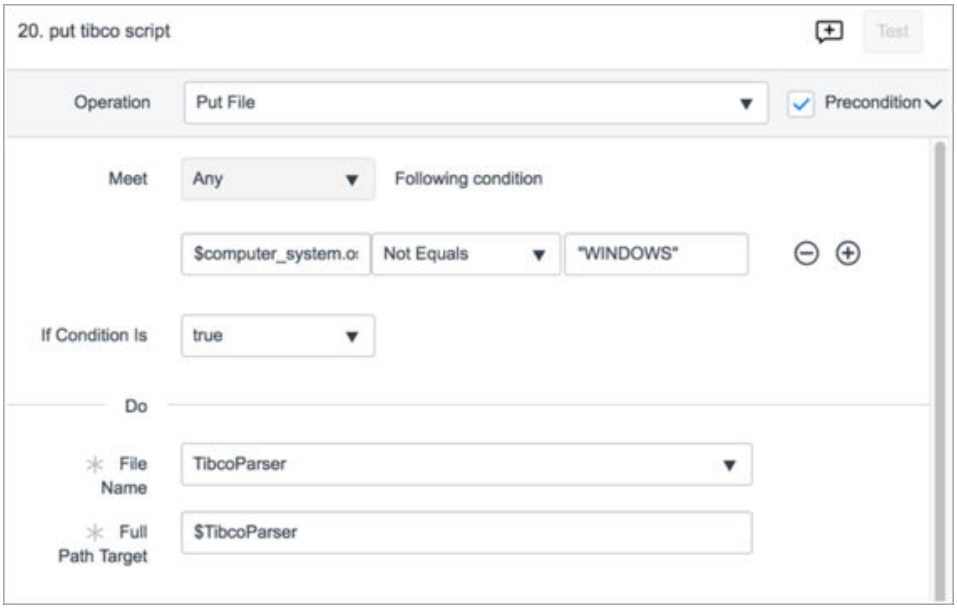
5. Specify the variable that refers to the MID Server path on the remote computer in the **Full Path Target** field.
6. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Application
CI Type	ActiveMatrix Business Works [cmdb_ci_appl_tibco_matrix]
Pattern	ActiveMatrix Business Works
Section	Tibco BW Identification
Step number and Name	20. put tibco script

Use the Put File operation to copy the TibcoParser file onto the target host and keep the full path to copied file in target host in the variable $TibcoParser.

Put File operation form



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Define an SNMP query

As part of creating or modifying a discovery pattern, you can use the **SNMP query** operation to execute SNMP queries on CIs that support the SNMP protocol, typically, load balancers.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. Select **SNMP query** from the **Operation** list.
2. To browse to the SNMP OID that you want to query using the debug mode:
   a. Activate pattern Debug mode.
   b. Click **Browse** and select the management information base (MIB) in the list or perform a search based on MIB values.
   c. Select either a single scalar value or multiple columns in the MIB tree.
   Multiple columns must be in the same table.
   d. (Optional) Click **Get Data** to display the data in the SNMP Browser form.
   e. Click **OK**.
3. To manually specify the SNMP OIDs for the query:
   - If the result of the SNMP query is a single string, choose **Scalar** and fill in the fields:

     Scalar fields

Field	Descriptions
SNMP OID	The SNMP OID to query.
Variable	Name of the scalar variable to hold the query result.

   - If the result of the SNMP query is a table, choose **Table**, click the plus icon, and fill in the fields:

Table fields

Field	Descriptions
Table name	Replace _Table_Name_ with the name of the Variables table.
Table OID	Replace _Table_OID_ with the SNMP OID to query.
New column name	Replace _new-column-name_ with the column name.
New OID	Replace _new_OID_ with the SNMP OID to query.

4. Select **Use Cache** to save the operation results in cache on the MID Server.
   Use cache to optimize discovery and avoid creating unnecessary load on central shared components, such as load balancers. The base system keeps operation results in cache for an hour.
5. Select **Terminate** to stop discovery if no results are found.
6. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

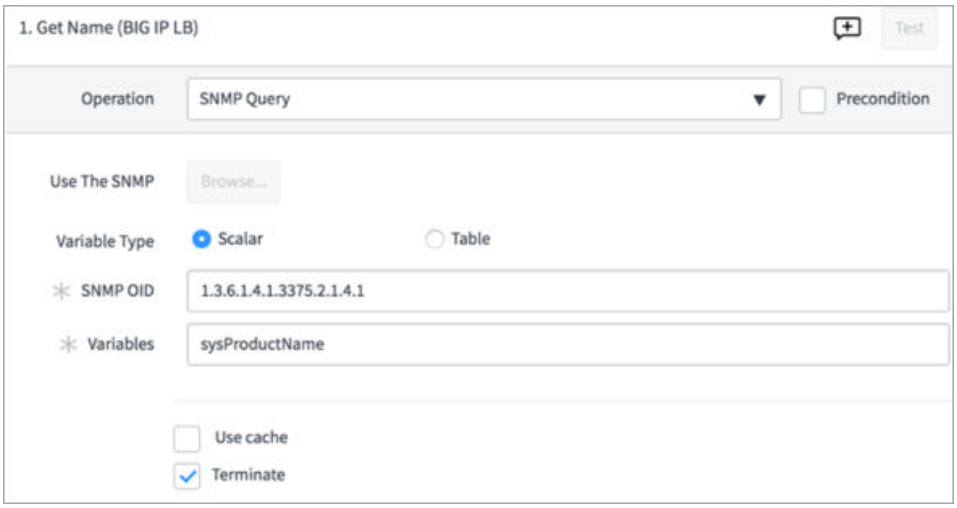
Example

The following configuration is an example of the **SNMP query** operation.

This operation is used in	This item
Hierarchy	Network > Load Balancer
CI Type	Load Balancer Service [cmdb_ci_lb_service]
Pattern	F5 BigIP LTM

This operation is used in	This item
Section	Identification for all entry point types
Step number and Name	1. Get Name (BIG IP LB)

The F5 BigIP LTM pattern uses the SNMP Query operation to extract the product name of the load balancer.



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related tasks

- Activate pattern Debug mode

Add a column to a table

As part of creating or modifying a discovery pattern, you can use the **Transform table** operation to add one or more computed columns to an existing table and place the results in a target table. The target table can be the source table.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

1. Select **Transform table** from the **Operation**.
2. Fill in the fields, as appropriate.

Field	Description
Source table	Specify the name of the source table.
Target table	Specify the name of the target table. The target table can be the same as the source table.

3. Click the plus icon to add each target field and fill in the fields, as appropriate.

Field	Description
Target Field Name	Specify a name for the column. It can be an existing column or a new one.
Value	Specify the operation expression that determines the values added to the column. You can use variables including values from tabular variables as described in Enter values and variables in patterns.

4. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

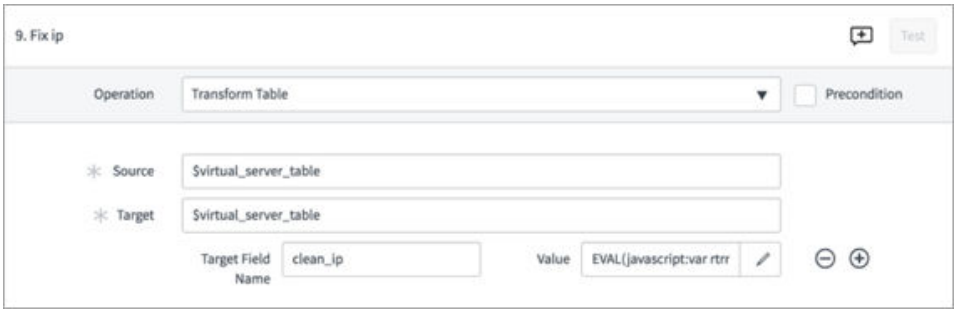
Example

The Transform Table operation is used to identify the Integration Feature Pack.

This operation is used in	This item
Hierarchy	Applications > Web Servers
CI Type	Load Balancer Service [cmdb_ci_lb_service]
Pattern	F5 BigIP LTM
Section	Identification for all entry point types
Step number and Name	9. Fix ip

For F5 load balancer, you want to translate IP addresses in the default hexadecimal format into decimal format. You use the Javascript that performs this translation in the value field. You add the "clean_ip" column containing IP addresses in decimal format to the existing (source) table.

Transform Table operation values



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related tasks

- Append two tables

Change credentials to default

As part of creating or modifying a discovery pattern, you can use the **Unchange user** operation to switch back to default administrative credentials if you previously changed them to non-default using the **Change user** operation.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Procedure

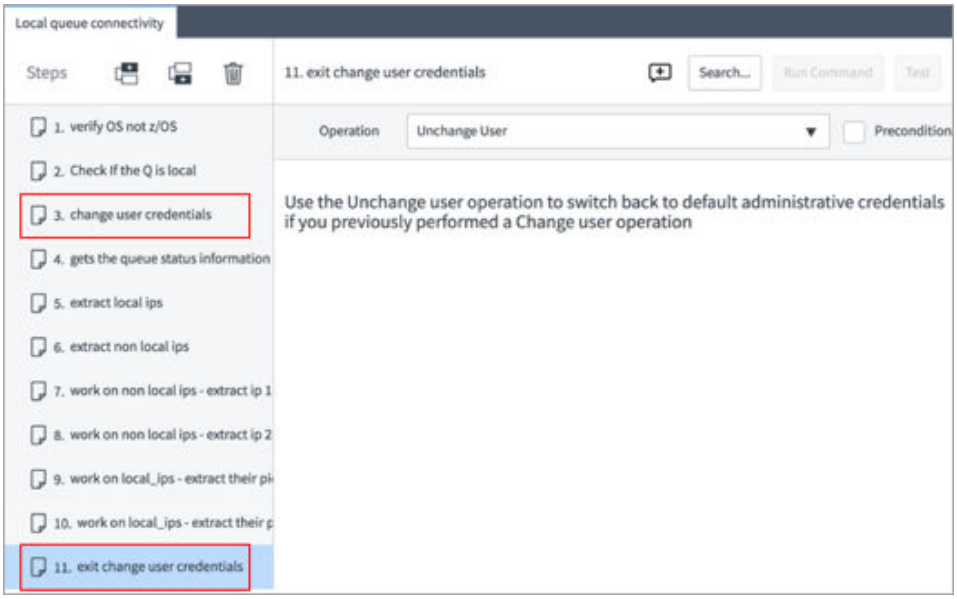
1. Select **Unchange User** from the **Operation** list.
2. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

Field	Value
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern
Section	Local queue connectivity
Step number and Name	11. exit change user credentials

As part of discovery of an IBM WebSphere MQ Queue, you change credentials to IBM WebSphere MQ. It allows you to manipulate local IPs to discover connections on the local queue. Once this segment of discovery is complete, you change credentials back to the default: IBM WebSphere MQ Queue credentials.

Unchange User operation form



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Related tasks

- Change credentials to non-default

Append two tables

As part of creating or modifying a discovery pattern, you can use the **Union table** operation to append two tables of the same format.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

Appending two tables adds the second table at the end of the first table and places the results in the additional, target table. If necessary, you can specify one of the source tables as the target table.

Procedure

1. Select **Union Tables** from the **Operation** list.
2. Fill in the fields, as appropriate.

Field	Description
First table	Specify the name of the first source table.
Second table	Specify the name of the second source table.
Target table	Specify the name of the table into which the results are placed.

3. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Application Servers
CI Type	WebSphere EAR
Pattern	J2EE EAR on Linux
Section	Oracle JDBC connectivity
Step number and Name	5. Union JDBCProviders with JDBCProviders2

A WebSphere EAR on Linux has two tables containing data on JDBC Providers. To add data from the second table at the bottom of the first table, use the **Union Tables** operation as follows:

What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Define WMI method invocation

As part of creating or modifying a discovery pattern, you can use the **WMI method invocation** operation to execute a method selected from a table returned by a WMI query.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Make sure that the step containing the WMI method invocation operation follows the step with the WMI query operation. The WMI query results in table which you must use as the source table for WMI method invocation operation.

Basic knowledge of programming is desirable.

About this task

This operation is relevant only for Windows.

Procedure

1. Select **WMI method invocation** from the **Operation** list.
2. Fill in the fields, as appropriate:

Field	Description
Enter Source Table	Specify the source table name. The source table must be the result of the WMI Query

Field	Description
	operation you perform before this step. You can enter a value from the specific field in a table as described in Enter values and variables in patterns.
Enter WMI Method	Select the desired method: • In the debug mode, click **Get methods** and select the method from the list. • Not in the debug mode, enter the name of the method as a string, for example, "RunDetails".
Enter Target Table	Specify the target table name.
Enter Target Column	Specify the name of the column to contain the results of the method invocation.

3. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Directory Services
CI Type	IIFP
Pattern	IIFP On Windows Pattern

This operation is used in	This item
Section	AD Home Forest connectivity stage-wmi
Step number and Name	2. invoke_wmi_method Run Details

During discovery of Identity Integration Feature Pack (IIFP), use the RunDetails WMI method to extract information on details from the ManagementAgents table. You discover this table earlier using the WMI query operation. In this case, the result is saved in the same table in the column named "details".

WMI Method Invocation

What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Define a WMI query

As part of creating or modifying a discovery pattern, you can use the **WMI query** operation to execute a query on a remote Windows system either explicitly or automatically. Successful values are logged in a specified target table.

Before you begin

Role required: pd_admin

Navigate to the relevant pattern step:

1. On the pattern form, select the relevant identification section for Discovery.

   Alternatively, select the relevant identification or connection section for Service Mapping.

2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

About this task

This operation is relevant only for Windows.

Procedure

1. Select **WMI Query** from the **Operation** list.
2. Define the namespace path in the **Namespace** field.
3. If in the debug mode:
   a. Click **Statement Accelerator** to populate the table and field values automatically.
   b. Select a table from the **Table name** pane.
   The field values are displayed in the **Fields** pane.
   c. Define parameters and operators for the condition in the **Add Condition Clause**.
   The actual query text is displayed in the **Get Full Statement**.
4. If not in the debug mode, enter the WMI query string in the **Get Full Statement**.
5. Enter the variable for the table in which the retrieved data is saved in the **Add Target Table** field.

You can enter a value from the specific field in a table as described in Enter values and variables in patterns.

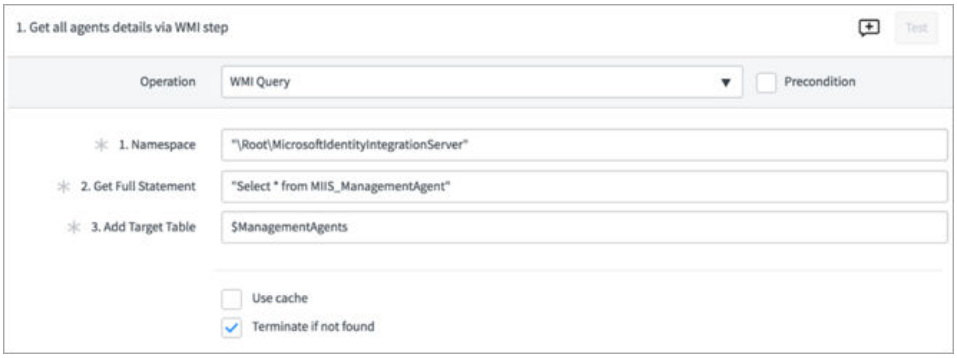
6. Select **Use Cache** to save the operation results in cache on the MID Server.
   Use cache to optimize discovery and avoid creating unnecessary load on central shared components, such as load balancers. The base system keeps operation results in cache for an hour.
7. Select **Terminate** to stop discovery if no results are found.
8. If in Debug mode, test the step by clicking **Test** and checking that the operation brings the result you expected.

Example

This operation is used in	This item
Hierarchy	Applications > Directory Services
CI Type	IIFP [cmdb_ci_directory_iifp]
Pattern	IIFP On Windows Pattern
Section	AD Home Forest connectivity stage-wmi
Step number and name	1. Get all agents details via WMI step

This step uses the WMI Query operation to extract data on agents from the MicrosoftIdentityIntegrationAgent namespace and save this data in the variable table named $ManagementAgents.

WMI Query operation form



What to do next

- Continue editing the pattern by adding a new step and defining its operation.
- Finalize the pattern.

Parsing strategies

In discovery patterns, you can use parsing strategies to analyze syntax of the source file. You extract values from parsed files, which allows you later to convert these values into variables.

There are several parsing strategies coming with the base system:

Parsing strategy	Description
Oracle LDAP file XML file INI file Properties file	Horizontal file parsing strategy (not vertical). You can use this parsing strategy only for text files. For more information, see Parse text from a horizontal file.

Parsing strategy	Description
JSON file (custom)	
Vertical File	Retrieve text from a structured text file where each set of data spans multiple lines. For more information, see Parse text from a vertical file.
After Keyword	Retrieve text directly following a specific keyword. For more information, see Parse text using keyword, command, and positional type.
Command Line Java Style	Retrieve the value of a command-line parameter using Java-style parameters. For more information, see Parse text using keyword, command, and positional type.
Command Line Unix Style	Retrieve the value of a command-line parameter using standard Unix parameters. For more information, see Parse text using keyword, command, and positional type.
Position From End	Retrieve text specified by its position from the end of the line. For more information, see Parse text using keyword, command, and positional type.

Parsing strategy	Description
Position From Start	Retrieve text specified by its position from the beginning of the line. For more information, see Parse text using keyword, command, and positional type.
Regular Expression	Retrieve text specified by a regular expression. This option requires familiarity with Regex Java syntax. For more information, see Parse text using a regular expression.
Delimited Text	Retrieve text specified by delimiters and position within the line (the most common way to retrieve text from generic text files). See Parse text using delimited text.

In addition to these parsing strategies, you can create custom parsing strategies to answer the needs of your organization.

Related tasks

- Customize parsing strategies

Parse text from a horizontal file

You can use the file type parsing strategy to parse text in files of the following formats: .ora file (used by various Oracle products), .properties file (common for Java), .xml file, and .ini file. For vertical files, use the vertical file parsing strategy instead.

Before you begin

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

You can use this parsing strategy only for text files.

> **Warning:** Do not use this parsing strategy for non-text files such as binary files.

You can define multiple extracts and variables. When identifying text for extraction into variables, what you are really doing is identifying the text location within a context.

You can use one of the following methods:

- In Debug mode, you can select the relevant string from the file contents in the text box. For each string you select, its position and delimiters relative to its context are stored. It enables the same definitions to apply to other files with the same structure even though the text varies. However, it selects the entire text within a context.

  For example, if you try to select only 456 in the text box of an XML file with the following line, the entire string between the keywords is selected.

  ```
  <ciTypeID>123-456-7890000000↙ciTypeID>
  ```

- On the Advanced Parsing Options form (outside of Debug mode), you can specify a delimiter and position to identify the text string. You can also use this form to make a more refined selection than from within the text box.

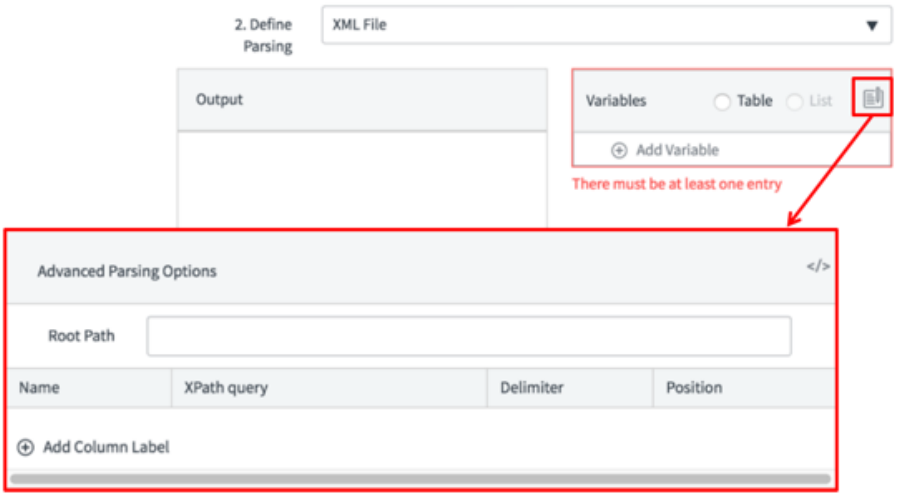
  For example, you could specify a delimiter (-) and the number of positions to extract after the delimiter (3) to extract the string (456).

Procedure

1. Select one of the parsing operations from the **Operation** list.
2. Select one of these options from the **Parsing strategy** list:
   - **Oracle**
   - **JSON file**
   - **Properties file**

- **XML file**
- **LDAP file**
- **INI file**

3. If working in the Debug mode, perform the following steps:
   a. Click **Retrieve** or **Retrieve File Content** to display the content of what you are parsing in the Output pane.
   b. Select the string in the text box.
   All matching strings in the same context are automatically selected.
   c. On the Define Variable Name form, assign the string to a variable by providing a unique and meaningful name and selecting **OK**.
   d. To identify additional strings and variables, click the plus icon.
4. Define the string to be parsed from within Debug mode, or on the Advanced Parsing Options form (outside of Debug mode).



Option	Description
Outside of Debug mode (Advanced Parsing Options form)	a. Click **Advanced** and specify the root path. The root path is the section (hierarchical

Option	Description
	branch in the file structure) where parsing takes place. b. Click the plus icon for each string and variable to be added and fill in the fields, as appropriate. • **Name**: Specify the column name. • **XPath query**: Specify the XPath query for the string. For example, appcmd/APP/@APP.NAME. • **Delimiter**: Specify the delimiter for the string. • **Position**: Specify the position of the string.

5. To end the discovery process if no results are found, select the **If not found** check box.
6. Click **Close Advanced**.

Parse text from a vertical file

In discovery patterns, you can use the **Vertical file** parsing strategy to parse text into variables for vertical files.

Before you begin

Basic knowledge of programming is desirable.

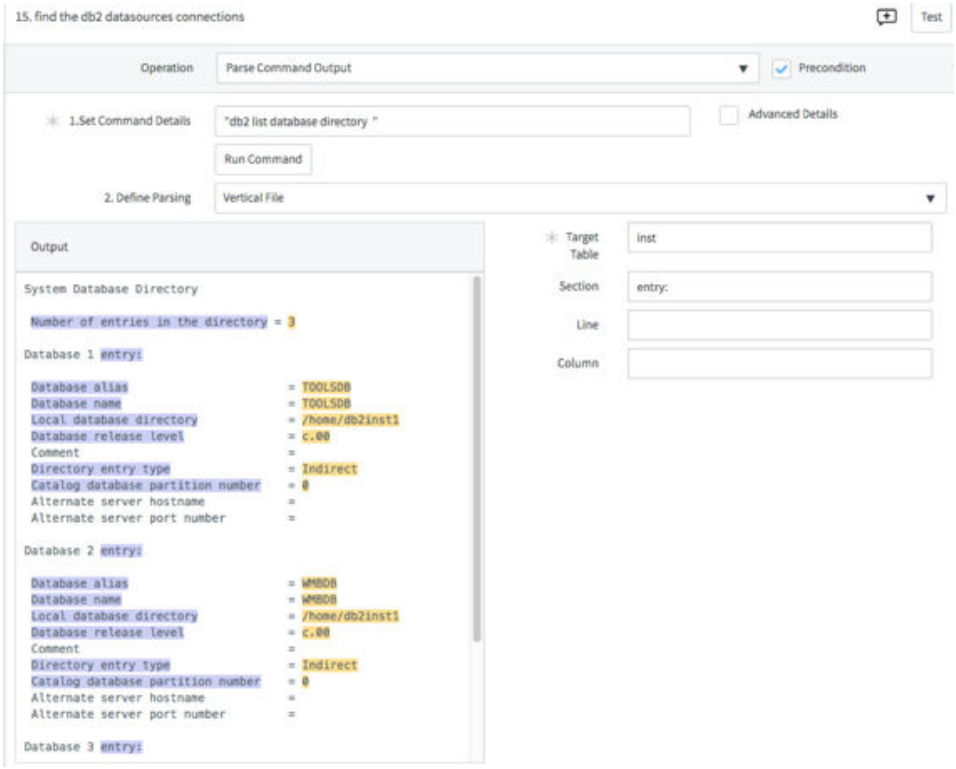
Role required: pd_admin

Procedure

1. Select one of parsing operations from the **Operation** list in one of the following locations:
   - the **Identification Sections** or **Connectivity Sections** for Service Mapping
   - the Step window for Discovery
2. Select the **Vertical file** from the **Parsing strategy** list.
3. In the **Target Table** field, enter the name of the table to hold variables extracted during this operation.
4. In the **Section** field, enter the title of the section.
   This parameter is used to identify the beginning of sections in the file.
5. In the **Line** field, enter the line number.
   The lines are counted from top to bottom.
6. In the **Column** field, enter the column number.
   The columns are counted from left to right.
7. To end the discovery process if no results are found, select the **If not found** check box.

Example

You can use the **Vertical File** parsing strategy for the Parse Command Output operation to extract information about a database. The section separator is **entry:**.



Parse text using keyword, command, and positional type

In discovery patterns, you can parse text into a variable for keyword, command, and positional type parsing strategies.

Before you begin

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

The following strategies are generally used to extract a value from a variable. You can use them to define only a single value for extraction. To extract multiple strings, define multiple steps.

- After keyword
- Command-line Java style
- Command-line Unix style
- Position from start
- Position from end

Procedure

1. Select one of the parsing operations from the **Operation** list in one of the following locations:
    - the **Identification Sections** or **Connectivity Sections** for Service Mapping
    - the Step window for Discovery
2. Select one of the keyword, command, or positional types from the **Define Parsing** list.
3. Click **Retrieve**.
   The content of the file is displayed in the **Output** pane.
4. Define the string to extract as a variable from within Debug mode:
    a. Select the string in the **Output** box.
       All settings are automatically entered in the variable fields.
    b. Provide a unique and meaningful name for the variable and select **OK**.
       The variable is added to the **Variables** table.
5. Define the string to extract as a variable outside of Debug mode:
    a. Specify the value and delimiter for the **Keyword**, **Position from Start**, or **Position from End**.

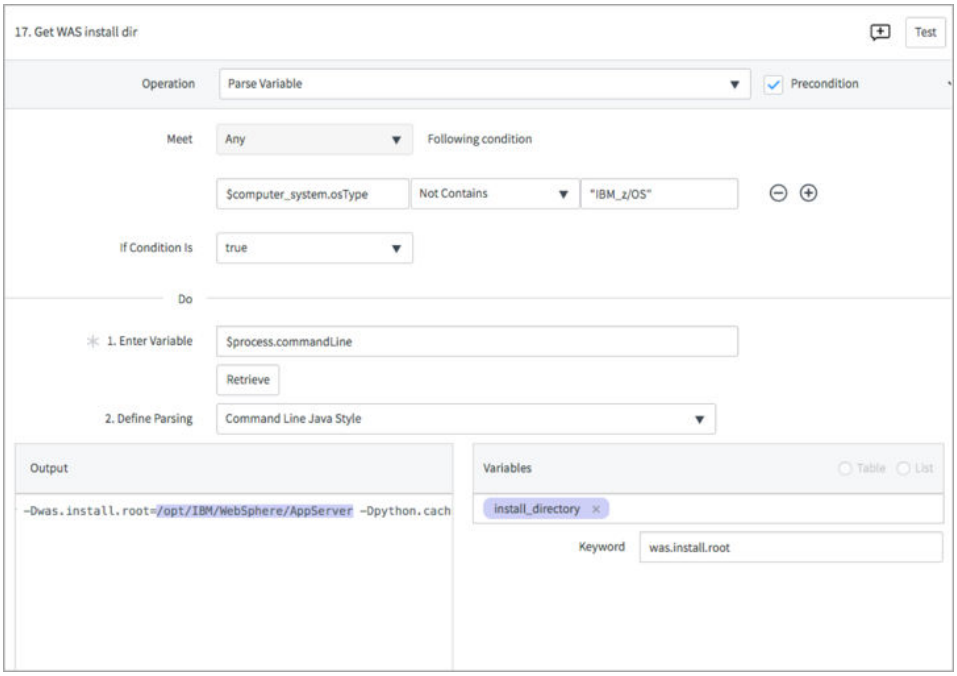
b. To add more strings and variables, select the plus icon.

6. To end the discovery process if no results are found, select the **If not found** check box.

Example

You can use the Command Line Java Style parsing strategy to extract the path of the installation directory of the WebSphere Server.

Parse Variable operation form



Parse text using a regular expression

In discovery patterns, you can parse text into variables using **Regular expressions** as the parsing strategy.

Before you begin

Familiarize yourself with the Regex Java syntax:

https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

- Navigate to the relevant pattern step:

  1. On the pattern form, select the relevant identification section for Discovery.

     Alternatively, select the relevant identification or connection section for Service Mapping.

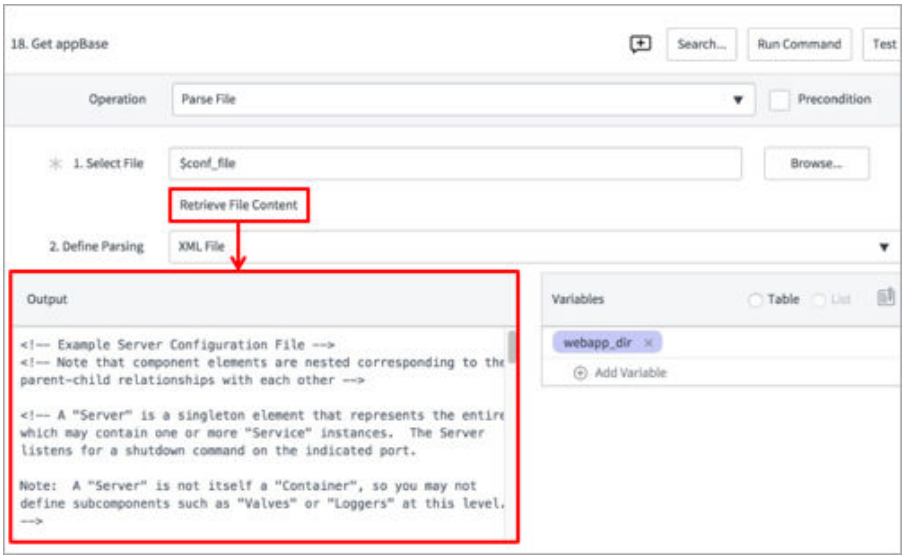
  2. Select the relevant pattern step or click  to add a step.

Basic knowledge of programming is desirable.

Role required: pd_admin

Procedure

1. Select one of the parsing operations from the **Operation** list.
2. Click **Retrieve** or **Retrieve File Content** to display the actual file in the **Output** pane.



3. Select the **Regular expression** from the **Define Parsing** list.
4. Fill in the fields, as appropriate.

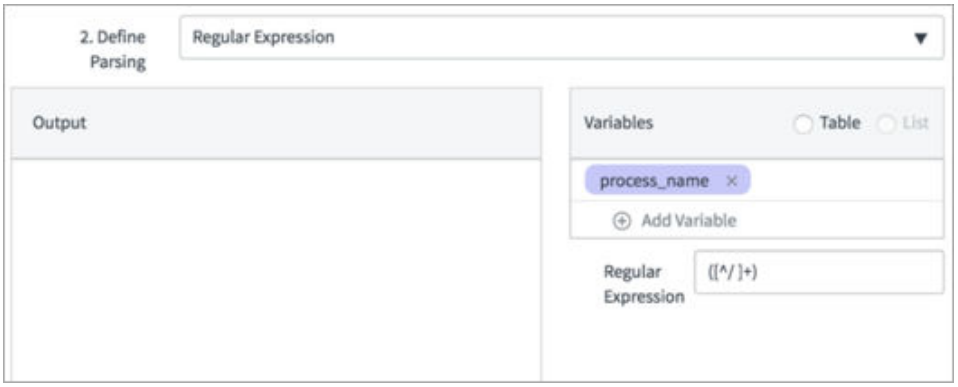
Field	Description
Regular expression	Specify the regular expression. You cannot select text in the text box using this parsing strategy. You can only specify a single expression. To extract multiple values, define several expressions enclosing each expression into parentheses. Matching of variables to parentheses occurs according to the order of the parentheses sets. (The first variable is

Field	Description
	matched to the first set of parentheses, and so on).
Set variables as	• **Table**: Select this option if the target table contains multiple columns. • **List**: Select this option for a single string (scalar). List must contain more than one variable.

5. To end the discovery process if no results are found, select the **If not found** check box.

Example

Regular expression operation form



Parse text using delimited text

In discovery patterns, you can parse text using the **Delimited text** parsing strategy.

Before you begin

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

You can use this parsing strategy for extracting data from any text file. Define which segment of this text file to extract by entering the symbol or word which serves as a boundary and defining the position.

Procedure

1. Select **Delimited text** from the **Define Parsing** list in one of the following locations:
   - the **Identification Sections** or **Connectivity Sections** for Service Mapping
   - the Step window for Discovery
2. Fill in the fields, as appropriate.

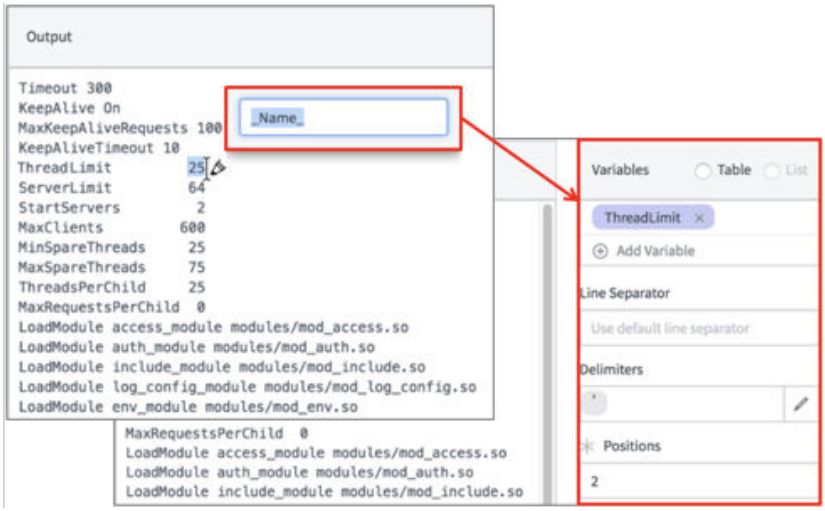
Field	Description
Include lines with	Specify the constraints used to determine which lines are included in the text selection.
Exclude lines with	Specify the constraints used to determine which lines are excluded in the text selection.
Line separator	Specify the non-default character used as a line separator in the text. Default separators are either NEWLINE or CARRIAGE RETURN, depending on the operating system.
Use default	Select this check box to use the default line separator.

3. To define parse the file and extract the string as a variable from within Debug mode:

   a. Select the string to be parsed in the text box.
   The name box appears.

   b. Provide a unique and meaningful name for the variable and press Enter.

   The variable is added to the **Variables** table. The Line Separator, Delimiters, and Positions attributes for the variable are filled in automatically.



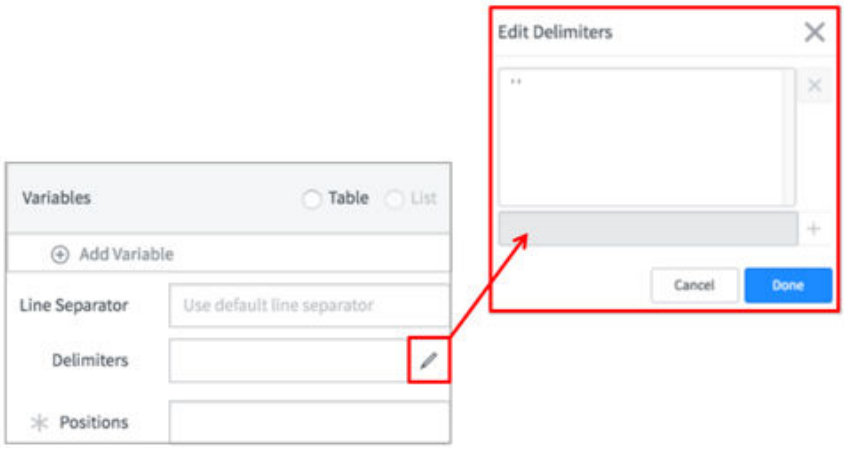
4. To define parse the file and extract the string as a variable outside Debug mode:

   a. In the **Variables** pane, click **Add Variable**.
   The new variable with the default name **_Name_** shows in the list of variables.

   b. Click the new variable and enter a unique and meaningful name.

   c. If necessary, enter a string in the **Line Separator** field.
   This string is used to identify where lines in the file end. For example, common line separators are "/n" or "/r".

d. Click the **Edit** icon next to **Delimiters**, enter the symbol or word to use as a delimiter and click the **Plus** icon.



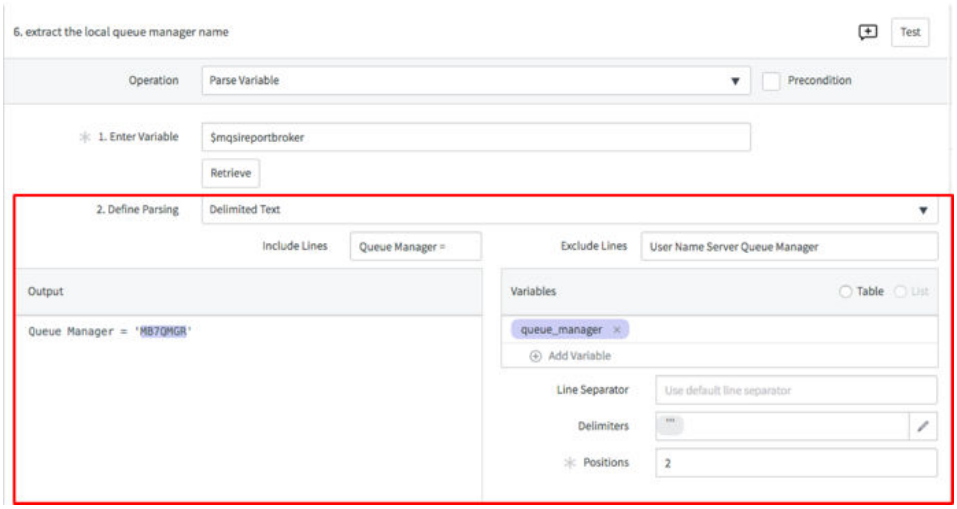
e. Specify the position.
For multiple positions, use commas as separators.

5. To end the discovery process if no results are found, select the **If not found** check box.

Example

You can use the Delimited Text parsing strategy for the Parse Variable operation to extract the Queue Manager name of an IBM WebSphere Message Broker.

Parse Variable operation form



Customize parsing strategies

You can modify parsing strategies that come as part of the base system or add new ones. Parsing strategies are part of discovery patterns, which Service Mapping and Discovery use to discover and map configuration items (CIs).

Before you begin

Practical knowledge of Java scripting is required.

Role required: pd_admin

About this task

Parsing serves to analyze syntax of the source file. You extract values from parsed files, which allows you to convert these values into variables. There are standard parsing strategies, which are part of the base system. If you cannot extract data using the standard parsing strategies, you can create your own parsing strategy. Custom parsing strategies are JavaScript files.

The JavaScript for parsing strategy must comply to the following guidelines:

- You may use the content variable to refer to the raw data resulting from executing the operation.
- Use rtrn to indicate the string, which is the result of your custom parsing.
- The script output must be in well-formed XML syntax.

  For example, the output must contain correct tags and markup:

```
<root>
<OS>Windows↙OS>
<version>10↙version>
↙root>
```

- If necessary, you may use third-party JavaScripts to convert the content into XML.

  For example, to convert the content from JSON into XML, use the X2JS JavaScript provided by open-source software.

```
var xtojs = new X2JS();
var result = xtojs.js2xml(jsonObj);
var finalResult = "<root>" + result + "↙root>";
```

Procedure

1. Navigate to **All > Pattern Designer > Custom Parsing Strategies**.
2. To modify the JSON custom parsing strategy, select JSON file. Alternatively, click **New** to create a new JSON file.
3. For the new custom parsing strategy, enter a name describing the new strategy in the **Name** field.
   For example, if the purpose of this strategy is to extract information using a certain protocol, use this protocol as the name.
4. Enter or modify the parsing strategy purpose in the **Description** field.
5. Write the JavaScript in the Script pane to define the business logic of parsing.
6. Click **Update** or **Submit**.

What to do next

Use this custom parsing strategy for defining operations inside pattern steps.

Related topics

- Syntax editor

Define the connection section

Define a set of discovery steps to discover outbound connections of a configuration item (CI). This operation is relevant only for patterns of the application type used by Service Mapping.

Before you begin

Create a pattern or select a pattern that you want to modify as described in Create or customize patterns.

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

A connection section identifies a type of outgoing connection. CIs can have multiple outgoing connections. Configure a separate connection section for each type of outgoing connection. For example, a .NET application CI can have outgoing connections of several types: ADO.NET, XML Web Services, or .NET. So, you must add connection sections for these three types to the .NET Application pattern.

Procedure

1. Navigate to **All > Pattern Designer > Discovery Patterns** and open the required pattern from the pattern list.

   The pattern must be of the Application type.

2. In the **Connectivity Section**, click **New**.

Note: The **Connectivity Section** is grayed out if the pattern cannot be edited for Service Mapping.

3. Enter the name, and click **Save**.
4. On the pattern form, click **Save** to save the pattern.
5. Click the created connection section.
   The pattern designer opens showing the connection step.
6. Select **Create Connection** from the **Operation** list.
7. Fill in the fields, as appropriate.

Field	Description
Select Connection Type	• **Application Flow**: Used between two applications (can be of the same type). • **Cluster**: Used for connections to CIs of the cluster type. Specify a cluster name. • **Inclusion**: Used for connections to an object that is included in the current object. For example, a connection from J2EE to EAR, and a connection from IIS to a website. • **Storage Flow**: Used for connections between configuration items (CIs) of host type and devices of storage type.
Select Entry Point	Select the entry point type from the list. For more information, see Entry point attributes.

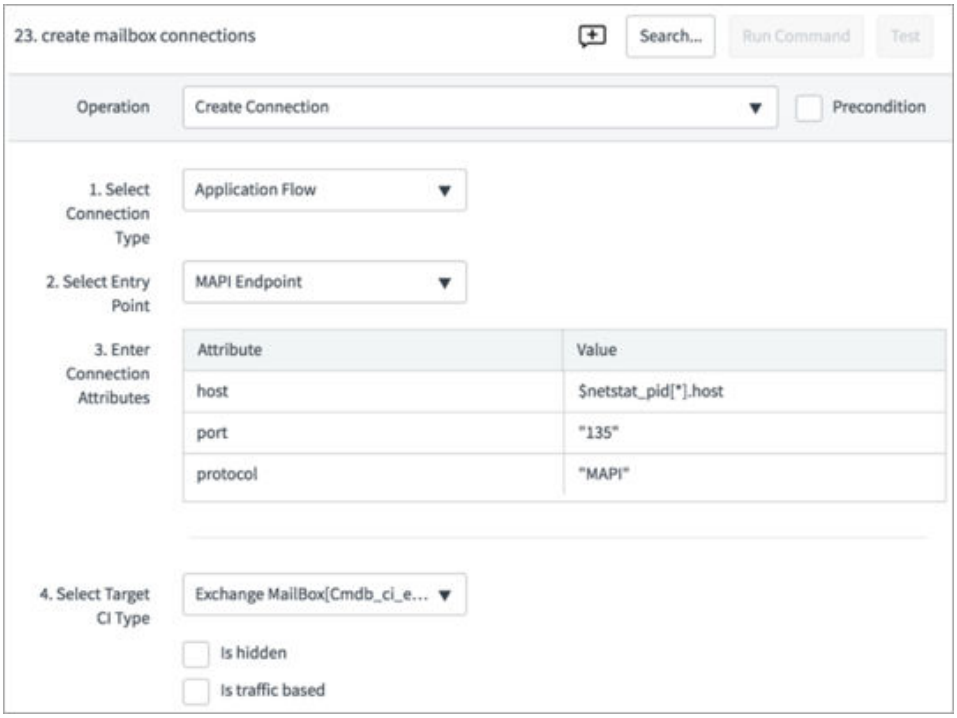
Field	Description
Enter Connection Attributes	Define attributes by either entering actual values or variables.
Select Target CI Type	Select the target CI type from the list.
Is hidden	Select the check box if the connection should be hidden (that is, not shown in the user interface), but is used for continuing the discovery flow.
Is traffic based	Select the check box to use the traffic-based discovery method for this connection.

8. If necessary, create more connectivity sections on the pattern form.

Example

This operation is used in	This item
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern
Section	Alias queues connectivity
Step number and Name	6. Create outgoing connection to alias queues

To create a connection from Microsoft Exchange CAS to Exchange Mailboxes, define the **Create Connection** operation as follows:



Example of creating an application pattern

Follow this example to see a step-by-step process of creating and defining the identification section for a new application pattern.

Before you begin

Basic knowledge of programming is desirable.

Role required: pd_admin

About this task

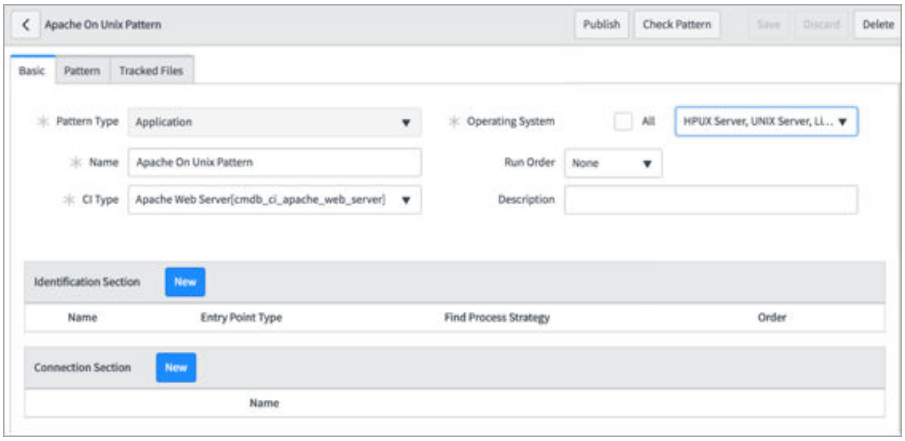
This example shows how to create a mapping pattern using the debug mode.

The pattern is for Apache Web Server on Unix.

Procedure

1. Navigate to **All > Pattern Designer > Discovery Patterns**.
2. Click **New**.
3. Define the basic pattern attributes as follows:

Field	Description
Pattern type	Select `Application`.
Name	Enter `Apache Web Server on Unix Pattern`.
CI Type	Select `Apache Web Server` from the list.
Operating system [Application patterns only]	Clear the **All** check box and select the following check boxes from the list: • `AIX Server` • `HPUX Server` • `Linux Server` • `Solaris Server` • `UNIX Server`
Run Order [Application patterns only]	Leave the default **None** setting.
Description	This pattern discovers Apache Web Servers on Unix versions up to 2.4.



4. Click **Save**.
5. Create the identification section and define its basic properties:
   a. Under **Identification Section**, click **New**.
   b. Configure the following parameters:

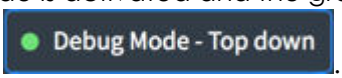
Field	Value
Name	Enter `Identification for HTTP(S) entry point type(s)`.
Entry Point Types [Application patterns only]	Select the following check boxes from the list: • `HTTP(S) Endpoint` • `TCP Endpoint`
Find Process Strategy [Application patterns only]	Select `Listening Port`.
Order	Enter `1`.

   c. Click **Save**.

6. Click the newly created identification section: **Identification for HTTP(S) entry point type(s)**.
   The new identification section opens on the separate tab.
7. Activate the debug mode:
   a. In the Pattern Designer, click **Debug Mode**.
      The Debug Identification Section window is displayed.
   b. Fill in the required details for the entry point type:

Field	Description
Debug Type	Select **Top down** for performing top-down discovery with Service Mapping or
Type	Select `HTTP(S)` for the entry point type from the list.
URL	Enter `http://10.196.39.244:6080/ITO`.

   c. Click **Connect**.

   The debug mode is activated and the green dot appears on the debug button: 

.

   Notice that the following variables are populated with values once the debug mode is active:

   - computer_system – the Apache host information
   - entry_point – identified by the URL in this case
   - process – the Apache process information

8. Check that the process name on the CI is Apache Web Server:
   a. Rename the first step of the identification section to `Check process name to match Apache`.
   b. Select `Match` from the **Operation** list.

c. Enter `$process.executable` in the first condition field.

d. Select `Contains` from the conditional operator list.

e. Enter `"httpd"` in the second condition field.

f. Click the plus icon to add another condition.

g. Enter `$process.executable` in the first condition field.

h. Select `Contains` from the conditional operator list.

i. Enter `"apache"` in the second field.

j. Define that this match operation must match one of these conditions: Select `Any` from the **Meet** list.

k. Click **Test** and verify that you get the following message: No changes were made during this test.

9. Populate the label attribute for your CI:

a. In the Steps tree, click  to add a step blow the first step.

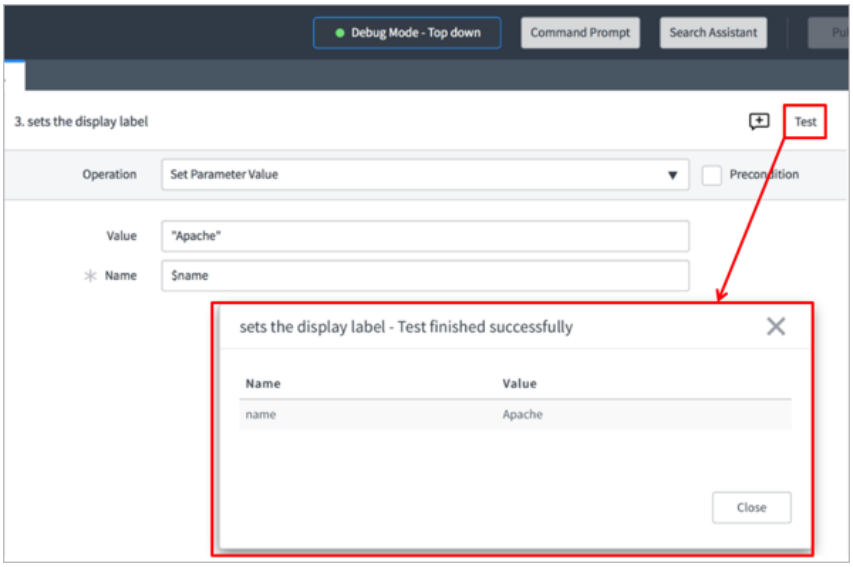
b. Rename the new step to `sets the display label`.

c. Select `Set Parameter Value` from the **Operation** list.

d. Enter `"Apache"` in the **Value** field.

e. Enter `$name` in the **Name** field.

f. Click **Test** and verify that the following message appears:



g. Click **Close**.

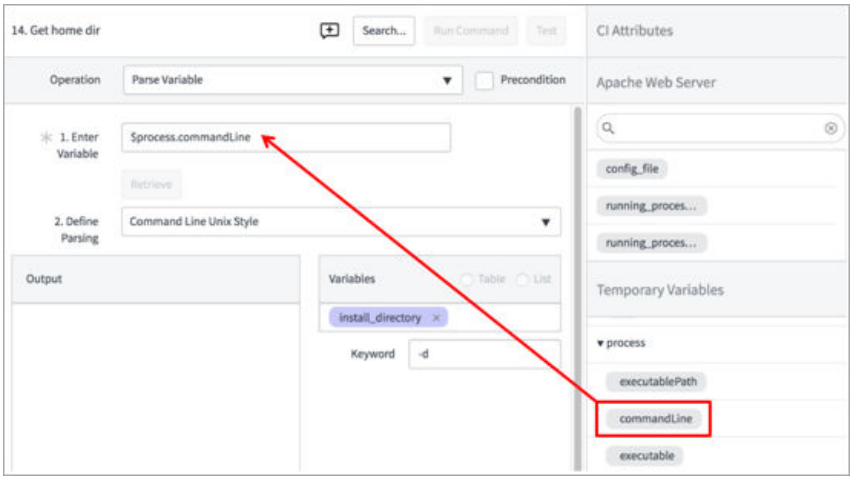
10. Populate the home directory attribute:

    a. In the **Steps** tree, add a step and rename it to `Get home dir`.

    b. Select `Parse variable` from the **Operation** list.
    This operation extracts the value after the -d in the content box.

    c. Expand the process variable in the **Temporary Variables** pane.

    d. Drag the commandLine variable from the Temporary Variables pane into the variable field under operation.



Note: For more information on using the drag-and-drop feature, see Enter values and variables in patterns.

e. Select Command line Unix style from the **Define Parsing** list.

f. In the **Variables** pane, add the new install_directory variable.

g. Click **Test**.

11. Obtain the home directory attribute from the HTTP daemon.
If the previous step populated the home directory attribute, skip this step. In this example, it is necessary to perform it.

a. In the **Steps** tree, add a new step and rename it to `Condition – check that home dir was set if not extract it from httpd –V`.

b. Select `Parse command output` from the **Operation** list.
This operation extracts the value after the -d in the content box.

c. Click **Precondition**.

d. Enter `$install_directory` in the condition value field.

e. Select `Is Empty` from the conditional operator list.

f. Select `True` from the **If condition Is** list.

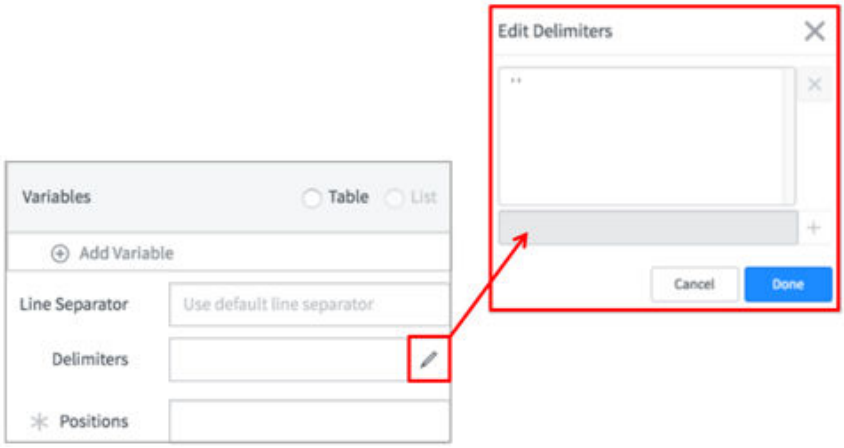
g. Enter `$process.executablePath+" -V "` in the **Set Command Details** field.

h. Click **Run Command**.

i. Select `Delimited text` from the **Define Parsing** list.

j. Enter `HTTPD_ROOT` in the **Include lines** field.

k. Click the **Edit** button next to **Delimiters**.



l. Add the two delimiters: equals (=) and quotes (").

m. Click **OK**.

n. Enter 2 in the **Positions** field.

o. Click **Test**.
The Debug Results window shows the home directory attribute populated with a value.

p. Click **OK**.

12. Populate the CI configuration file attribute:

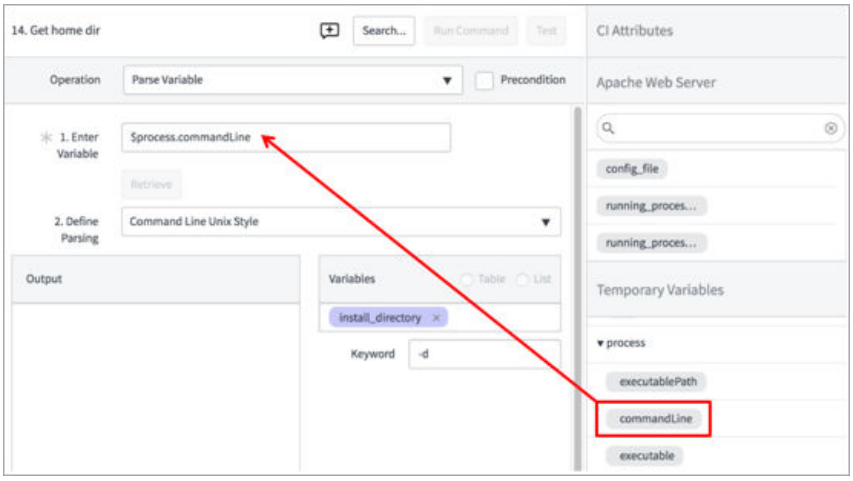
a. In the **Steps** tree, add a new step and rename it to `Get config file`

b. Select `Parse variable` from the **Operation** list.

c. Expand the process variable in the **Temporary Variables** pane.

d. Drag the commandLine variable from the **Temporary Variables** pane into the **Enter Variable** field.
The value is populated: /usr/sbin/httpd2-prefork -f /etc/apache2/httpd.conf.

e. Select `Command Line Unix Style` from **Define Parsing** list.



f. Click **Retrieve**.

g. Enter `-d` in the **Keyword** field.

h. Select the value in the **Output** pane and create a new variable named install_directory.

i. Click **Test**.

13. Extract the CI configuration file attribute from the HTTP daemon:
If the previous step populated the configuration file attribute, skip this step. In this example, it is necessary to perform it.

a. In the **Steps** tree, add a new step and rename it to `Condition – check that conf_file was set if not extract it from httpd –V`.

b. Select `Parse command output` from the **Operation** list.

c. Click **Precondition**.

d. Enter `$config_file` in the condition value field.

e. Select `Is Empty` from the conditional operator list.

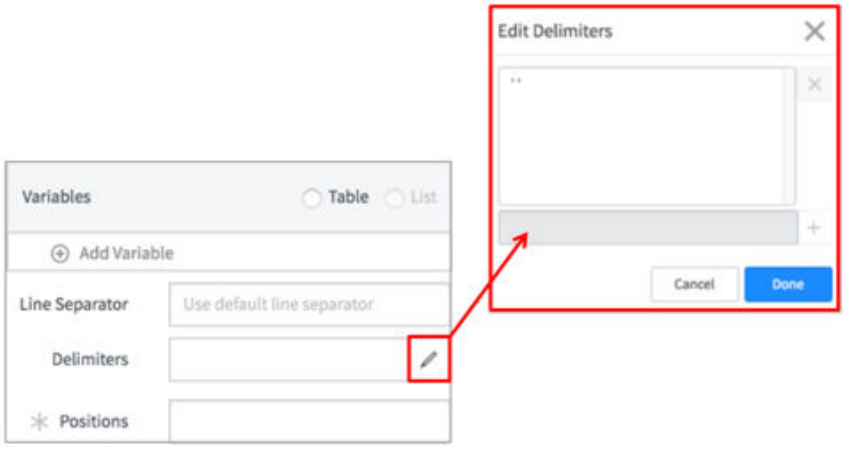
f. Enter `$process.executablePath+" -V "` in the **Set Command Details** field.

g. Click **Run Command**.

h. Select `Delimited text` from the **Define Parsing** list.

i. Enter `SERVER_CONFIG_FILE` in the **Include lines** field.

j. Click the **Edit** button next to **Delimiters**.



k. Add the two delimiters: equals (=) and quotes (").

l. Click **OK**.

m. Enter 2 in the **Positions** field.

n. If the new conf_file variable is not added automatically, create it in the **Variables** pane.

o. Click **Test**.
The Debug Results window shows the configuration file attribute populated with a value.

p. Click **OK**.

14. If the configuration file attribute is still not populated, perform this step:

a. In the **Steps** tree, add a new step and rename it to `default location of conf file`.

b. Select `Set Parameter Value` from the **Operation** list.

c. Click **Precondition**.

d. Enter `$conf_file` in the condition value field.

e. Select `Is Empty` from the conditional operator list.

f. Select `True` from the **If condition is** list.

g. Enter `$home_dir+"/conf/httpd.conf"` in the **Value** field.

h. Enter `$conf_file` in the **Name** field.

i. Click **Test** and verify that the configuration file attribute is populated.

15. Concatenate the home directory and configuration file values:

a. In the **Steps** tree, add a new step and rename it to `check if the SERVER_CONFIG_FILE is relative or not`.

b. Select `Set Parameter Value` from the **Operation** list.

c. Click **Precondition**.

d. Enter `$conf_file` in the condition value field.

e. Select `Starts With` from the conditional operator list.

f. Enter `"/"` in the string value.

g. Select `False` from the **If condition is** list.

h. Enter `$home_dir+"/"$conf_file` in the **Value** field.

i. Enter `$conf_file` in the **Name** field.

j. Click **Test** and verify that the configuration file attribute is populated.

16. Populate the version attribute:

a. In the **Steps** tree, add a new step and rename it to `get version from version.signature (IBM HTTPSERVER)`.

b. Select `Parse file` from the **Operation** list.

c. Enter the concatenated `$install_directory` variable and `"/version.signature"` string ($home_dir+"/version.signature") in the **Select File** field.

d. Click **Retrieve File Content**.

e. Create the Version variable in the **Variables** pane.

f. Click **Test** and verify that the version attribute is populated. In this example, the version is not extracted at this stage.

17. Extract the version attribute from the HTTP daemon:

a. In the **Steps** tree, add a step and rename it to `Condition - check that version was set if not extract it from httpd -v`.

b. Select `Parse command output` from the **Operation** list.

c. Click **Precondition**.

d. Enter `$version` in the condition value field.

e. Select `Is Empty` from the conditional operator list.

f. Click the plus icon (**+**) to add another condition.

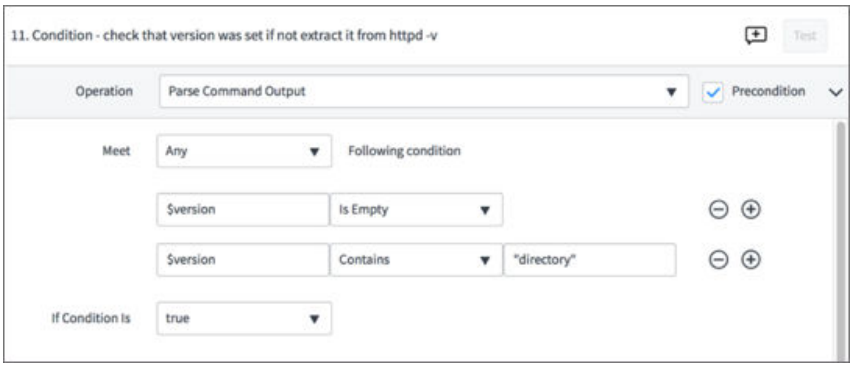
g. Enter `$version` in the condition value field.

h. Select `Contains` from the conditional operator list.

i. Enter `"directory"` in the string value.

j. Select `Any` from the **Meet** list.

k. Select `true` from the **If precondition is** list.

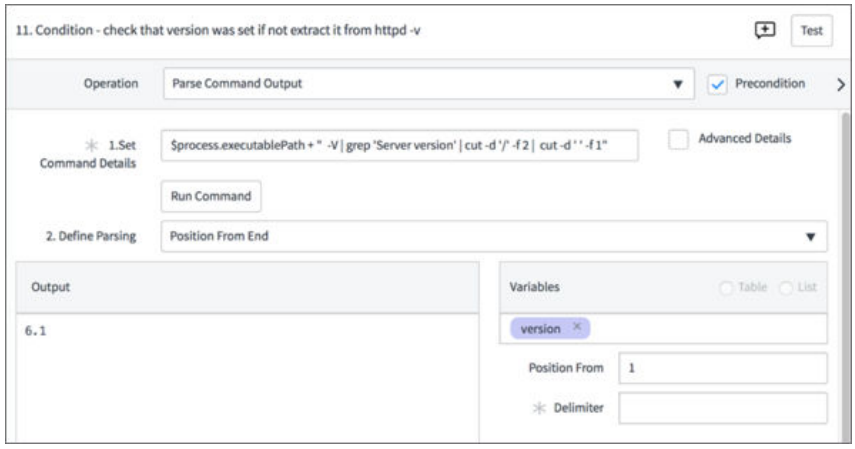


l. Enter `$process.executablePath+" -V | grep 'Server version' | cut -d '/' -f 2 | cut -d ' ' -f 1"` in the **Set Command Details** field.

m. Click **Run Command** and verify that the version attribute appears in the **Output** pane.

n. Select `Position From End` from the **Define Parsing** list.

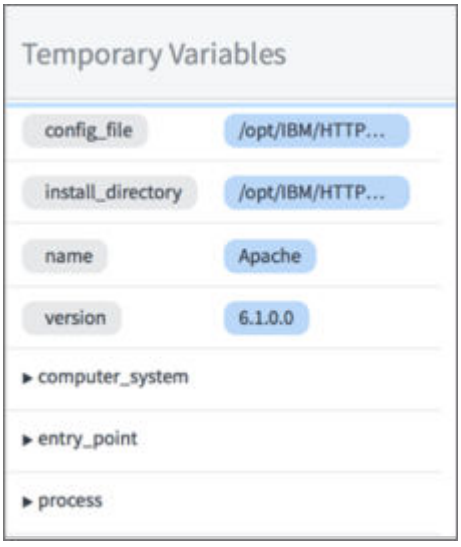
o. Enter `1` in the **Positions** field in the **Variables** pane.



p. Click **Test** and verify that the configuration file attribute is populated.

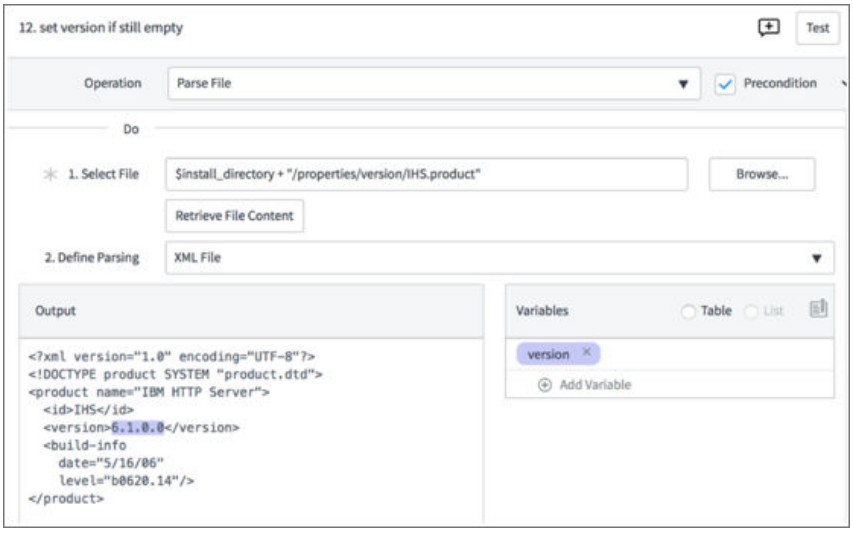
Note:

The version number appears only in the **Temporary Variables** pane, not in the **CI Attributes** pane.



Notice that at this stage you have successfully identified the Apache Web Server and populated its various attributes except the version attribute that is left blank on purpose.

18. If the version is still not populated, extract it from the IHS.product file:
    a. In the **Steps** tree, add a step and rename it to `set version if still empty`.
    b. Select `Parse File` from the **Operation** list.
    c. Click **Precondition**.
    d. Enter `$version` in the condition value field.
    e. Select `Is Empty` from the conditional operator list.
    f. Click the plus icon (**+**) to add another condition.
    g. Enter `$version` in the condition value field.
    h. Select `Contains` from the conditional operator list.
    i. Enter `"directory"` in the string value.
    j. Select `Any` from the **Meet** list.
    k. Select `true` from the **If precondition is** list.
    l. Enter `$install_directory + "/properties/version/IHS.product"` in the **Select File** field.
    m. Click **Retrieve File Content**.
    The content of the file is displayed in the **Output** pane.
    n. Select **XML File** from the **Define Parsing** list.
    o. Mark the version value in the **Output** pane, and then enter `version` in the variable pop-up.

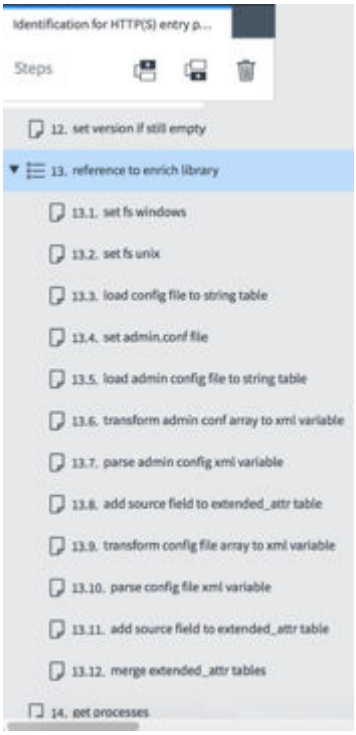


19. Find additional attributes by reusing a shared step library.

    a. In the **Steps** tree, add a step, and rename the new step to `reference to enrich library`.

    b. Select `Library reference` from the **Operation** list.

    c. Select `Apache Enrich Attributes` from the **Library** list.
    This operation inserts a sequence of preconfigured substeps into the step tree. For information on how to create shared step libraries, see Reuse a shared step library.



20. Populate process-related attributes:
    a. In the **Steps** tree, add a step and rename it to `get processes`.
    b. Select `Get process` from the **Operation** list.
    c. Enter `"httpd"` in the **Command Line** field.
    d. Enter in the `$procs` in the **Specify Target Variable** field.
    e. Click **Test** and verify that the attributes are displayed:

f. Click **Close** when done.

21. Discover the process IDs:

    a. In the **Steps** tree, add a step and rename it to `set process_ids`.

    b. Select `Parse variable` from the **Operation** list.

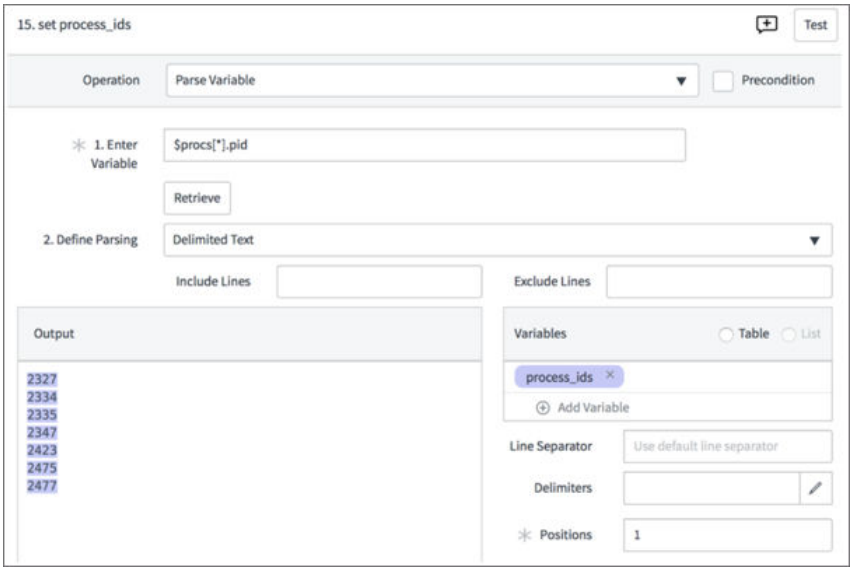
    c. Enter `$procs[*].pid` in the **Enter Variable** field.

    d. Select `Delimited text` from the **Define Parsing** list.

    e. Click **Retrieve**.
    The content is displayed in the **Output** pane.

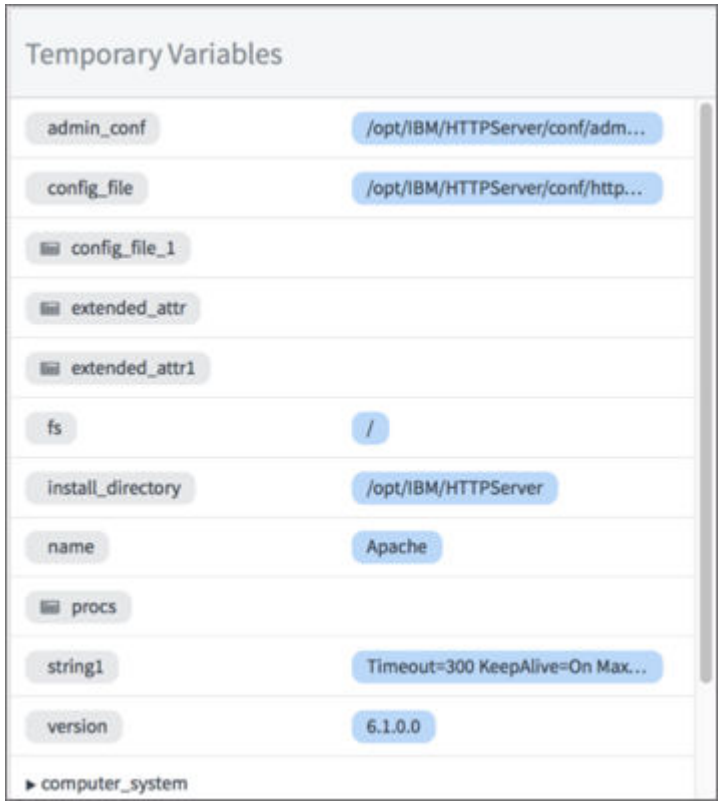
    f. Create the process_ids variable in the **Variables** pane.

    g. Enter `1` in the **Positions** field.

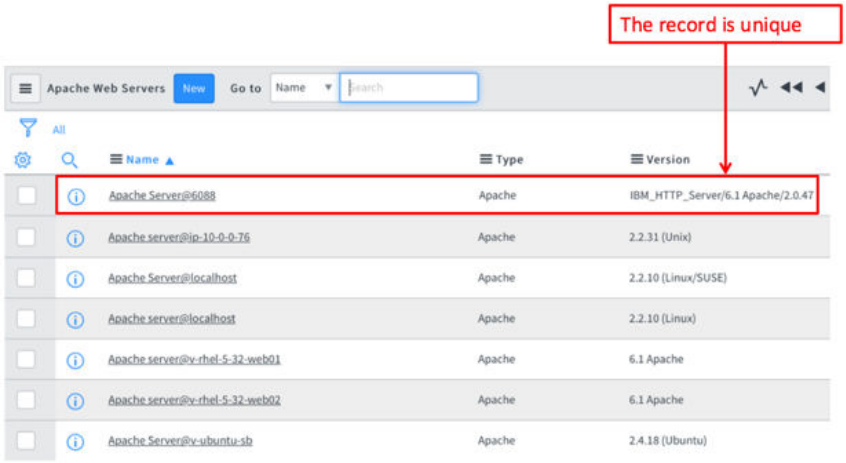


h. Click **Test**.

i. Verify that all the necessary attributes are populated:



22. Click **Save**.
23. Verify that Discovery and Service Mapping can use the identification section you defined:
    a. Run the horizontal discovery of an Apache Web Server using this pattern.
    b. Navigate to the table of Apache Web Servers and check that there is an entry for this CI.
    c. Run the top-down discovery of the same CI.
    d. Check the same Apache Web Servers table.
    e. Verify that the record is not duplicated.



It means that the results of both horizontal and top-down discovery are written in the CMDB under the same record. The identification section of the pattern is correct.

Related tasks

- Activate pattern Debug mode