ServiceWatch

Customization Guide

Version 3.6

August 2014

Contact Information

For customer support:
http://www.servicenow.com/support/contact-support.html

For all other purposes:
http://info.servicenow.com/contact-us

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

Chapter 1: Introduction

This chapter contains the following topics:

About This Guide

The ServiceWatch Customization Guide is intended to help you define discovery patterns for your in-house software and for proprietary 3rd party software that ServiceWatch does not yet support.

This guide assumes that you are familiar with the information in the ServiceWatch User Guide, especially the concepts and usage of Entry Point, Configuration Items and Discovery patterns.

Before defining discovery patterns for applications, you need first to define entry point types and configuration items needed for those applications.

Where to Go From Here

The chapters in this guide are arranged in the order that you should use them.

- First, define needed entry point types. For instructions, see CHAPTER 2: CONFIGURING ENTRY POINT TYPES.
- Next, define needed configuration item types. For instructions, see CHAPTER 3: CONFIGURING CI TYPES.
- Finally, define needed Discovery Patterns. Instructions for performing this task are divided into two chapters. For each Discovery Pattern that you need to define, you should:
  - ✓ First, configure the Discovery Patterns. For instructions, see CHAPTER 4: CONFIGURING DISCOVERY PATTERNS.
  - ✓ Then, define the detailed steps of the discovery pattern. For instructions, see CHAPTER 5: DEFINING DISCOVERY STEP DETAILS.

Chapter 2: Configuring Entry Point Types

This chapter contains the following topics:

Concepts

All objects in the business service topology contain an entry point (and, except at the lowest level, outgoing connections). Each entry point is characterized by an entry point type, which in turn determines the attributes of the entry point. ServiceWatch comes with a pre-supplied list of entry point types, each having a particular set of attributes defined for it.

You can optionally define default values for these attributes. Thus, when you define an initial entry point for a new business service and select its entry point type (as described in the *ServiceNow ServiceWatch User Guide*), the attributes for the entry point type will be displayed, along with any default values you defined.

You can also add, edit, or delete entry point type attributes; and you can create new entry point types. You perform all entry point type definition modifications via the Knowledge Base facility.

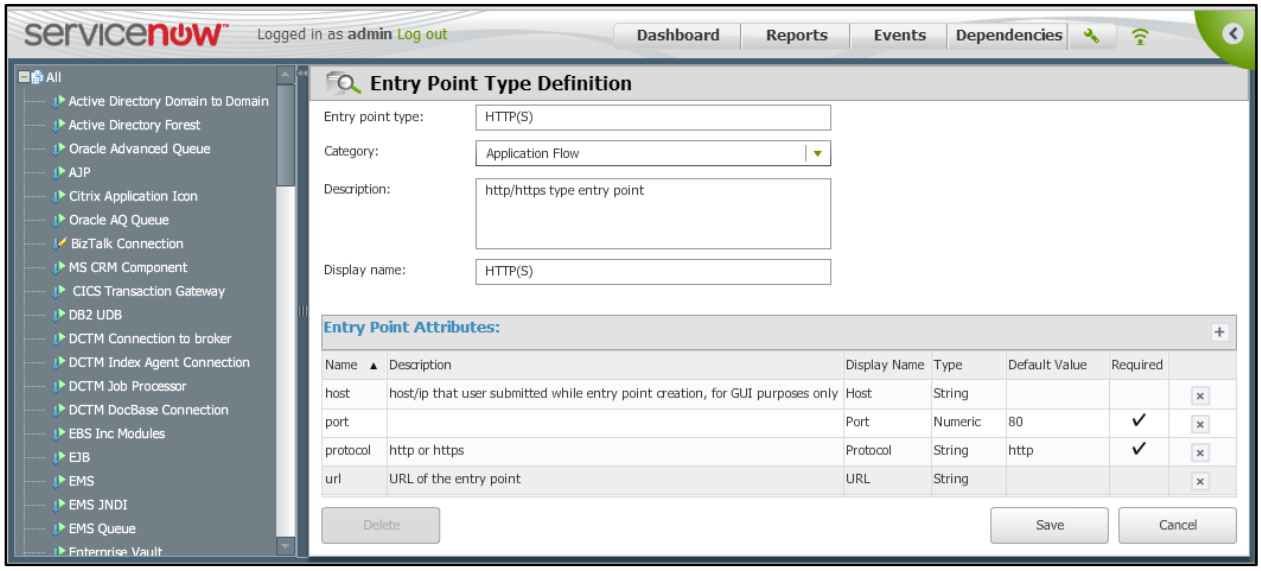
Modifying Entry Point Type Definitions

To modify an entry point type definition:

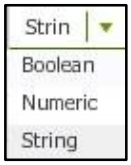
1. Click  in the Menu Bar.
2. In the list of **Settings** options, click the **Entry Point Types** link.
3. In the **All** [Entry Points] tree in the left pane, select the node of the entry point type you want to edit.
   The **Entry Point Type Definition** screen (Warning: Do ***not*** click the browser back button  . Doing so will take you out of the ServiceWatch application.
4. Figure 1) displays the details for the selected entry point type.
5. Edit the fields in the top half of the screen.
   - ✓ The name of the **Entry point Type** is mandatory.
   - ✓ Click the **Category** down arrow to display the two valid values:
     - ➢ **Application Flow** – Connections between two applications (that is, between one application and another (even of the same type). This value is appropriate for most Entry Point Type definitions.
     - ➢ **Inclusion** – One object is included in the other (for example, IIS Website, J2EE EAR).
   - ✓ **Display name** is the name that appears in screens that require selection of the entry point (for example, the **Entry Point** panel in the **Business Service definition** wizard).

   Warning: Do ***not*** click the browser back button  . Doing so will take you out of the ServiceWatch application.

Figure 1: Entry Point Type Definition screen



6. Edit the **Entry Point Attributes**.
   - ✓ You can specify a default value for any attribute. This value will appear any time the attributes are displayed; however, they can be edited in other definition screens.
   - ✓ You can edit an attribute's name, description, display name, and type. When you click the Type value, a down-arrow appears. Click that arrow to display a drop-down list that enables you to select a different Type.



   - ✓ To delete an attribute, click the [×] in the last column of its row in the **Entry Point Attributes** table. All entry point types *must* have a host attribute; so do not delete this attribute.
   - ✓ To add an attribute, click [+] in the **Entry Point Attributes** table header. The next unused row in the table is highlighted. Enter the attribute name and values in that row.
7. When done, click [Save], or [Cancel] to exit without saving, or [Delete] to delete an existing entry point type.

Adding a New Entry Point Type

To add a new entry point type:

1. Right-click the **All** root of the configuration tree of the **Entry Point Types** screen. In the pop-up menu, select **Add New Entry Point Type**.

   Note: As an alternative, you can right-click an existing entry point type, select the **Clone Entry Point Type** option, and modify the selected entry point's definition as required.

2. Fill in the details for the new entry point type. For instructions, see MODIFYING ENTRY POINT TYPE DEFINITIONS. Be sure to define a host attribute for the new entry point type.
3. Click [Save], or [Cancel] to exit without saving, or [Delete] to delete an existing entry point type.

Chapter 3: Configuring CI Types

This chapter contains the following sections:

Concepts

When you define an entry point for a business service, you identify the host computer of the top level application in the business service.

Discovery can go to that host, but it cannot automatically know which application on the host is the starting point, nor can it know what connections are outgoing from that application. Yet, these are the very tasks that discovery must perform.

Therefore, to enable discovery, it is necessary that the following information is defined:

- Configuration Item (CI) types – these are types of applications (for example, Apache Server, WebSphere) found in the business service.
- Discovery patterns – these are patterns that discovery will use to identify both the application and its outgoing connections.

ServiceWatch comes with a set of predefined set of CI types and their related discovery patterns. CI type definitions are spread among predefined categories (and in some cases, sub-categories) in the configuration tree. The top level categories containing CI types are:

- Applications
- Network
- Storage
- Virtualization

Discovery patterns for a CI type appear in the configuration tree under the CI type definition.

Although ServiceWatch comes with predefined CI types and discovery patterns for a large number of applications, your site might use in-house applications, or off-the-shelf applications for which no CI types are currently defined. In such cases, you will need to define the relevant CI types and discovery patterns.

You can also customize predefined CI types.

This chapter explains how to define and customize CI types.

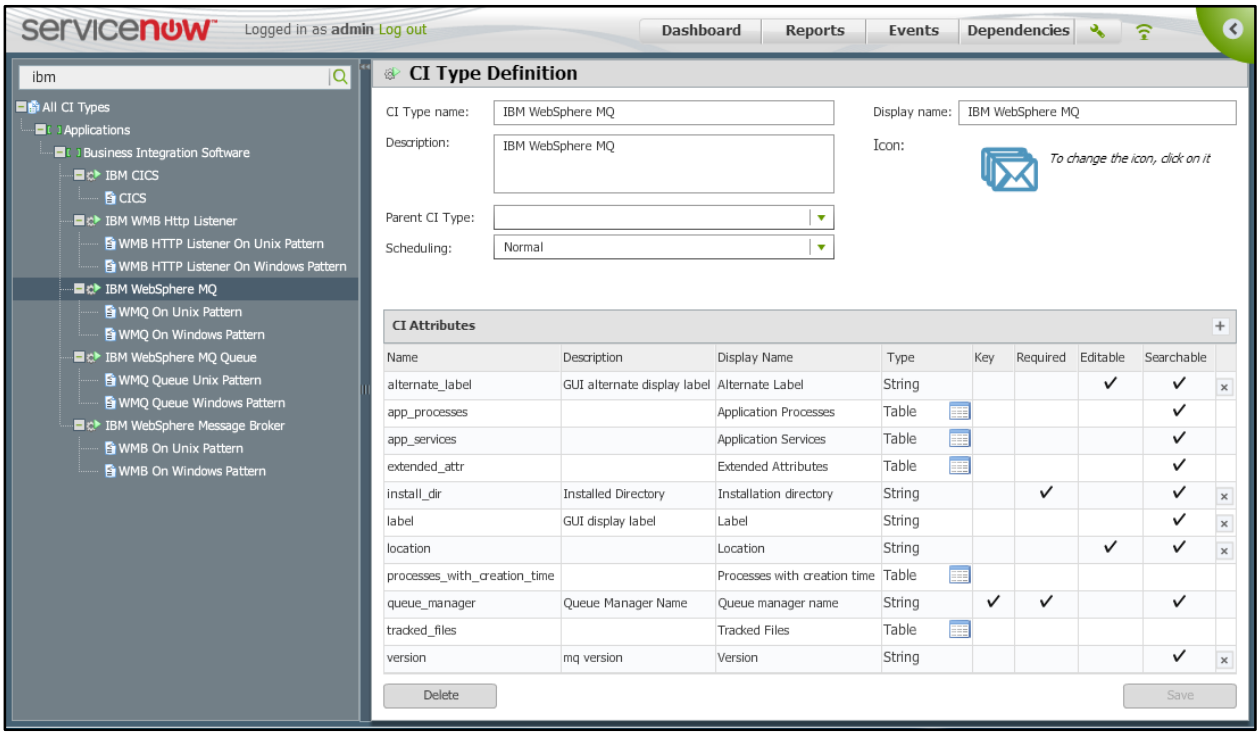
Defining CI Types

1. Click  in the Menu Bar. In the list of **Settings** options, click the **CI Types, Patterns & Monitors Definitions** link. Expand the relevant category and subcategory nodes in the tree.
2. Do either of the following in the **All CI Types** tree:

- ✓ To add a new CI Type, right click the relevant category or subcategory, and in the pop-up menu select **Clone CI Type**. When the selected CI type definition is displayed, modify its definition as required.
- ✓ To edit an existing CI Type, click the node of that CI Type. You can filter the nodes that are displayed by typing a (case insensitive) string in the search box above the **All CI Types** tree.

The CI Type Definition screen is displayed. FIGURE 2 illustrates a complete CI Type Definition.

Figure 2: CI Type Definition screen

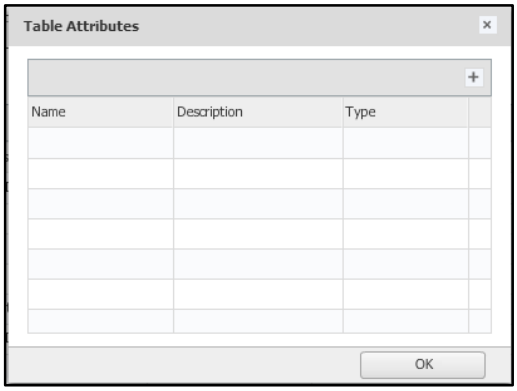


Fill in or modify the values in the screen as described in the following steps.

3. Fill in or edit the fields in the top half of the screen.
   - ✓ **CI Type name** (mandatory) is the actual name you are assigning to the CI type. If you want a different (for example, abbreviated) name value to appear in the Properties pane of the Map screen, specify the alternative name in the **Display name** field (optional).
   - ✓ **Description** is optional.
   - ✓ **Icon** – a default icon appears. To change the icon, click the icon, and then in the **Choose an Icon** dialog box, click the icon you want to use.
   - ✓ **Parent CI Type** – If the CI Type is always included in another CI type, select its parent CI type from the drop-down list. For example, because Website is always included in IIS, IIS is the parent CI type for Website.
   - ✓ **Scheduling** – Select Frequent, Normal or Rare from the Scheduling drop-down list.
4. Add (or edit) the **CI Attributes** as follows:
   a. To add an attribute, click  in the **CI Attributes** header. A blank entry line is added.
   b. Specify the attribute's **Name**, **Description**, and **Display Name**.

c. Click in the **Type** field and click its down arrow. Its drop-down list is displayed. Select the attribute Type (Table, Boolean, Numeric, String).

d. If you select **Table** as the attribute type, a table icon ( ) is displayed in the **Type** field. Click the table icon to open the **Table Attributes** dialog box and fill in the table details as follows:



i. To add a table attribute, click + in the table header. The default value **String** is inserted in the **Type** column of the first blank entry line in the table.

ii. Fill in a **Name** and **Description** for the table attribute.

iii. Repeat steps i and ii as needed.

iv. To delete a table attribute, click × on that attribute's row.

v. When done, click OK in the **Table Attributes** dialog box.

e. Select the **Key**, **Required**, and/or **Editable** cells and select (check) or clear their checkboxes:

i. If the attribute will be needed (alone or with other attributes) to identify an object of this CI type, select **Key**.

ii. If the attribute must have a defined value, select **Required**. If discovery does not find a value for a required attribute for an object, it issues an error message.

iii. If the attribute should be editable in the **Properties** pane in the **Map** screen, select **Editable**. (Attributes whose Type value is **Table** cannot be editable.)

f. Repeat step 4 for each attribute to be added.

5. To delete an attribute, click the × icon on that attribute's row.

6. When you finish defining the CI Type, click Save.

Chapter 4: Configuring Discovery Patterns

This chapter contains the following topics:

Concepts

Discovery patterns are patterns that discovery uses to help it identify entry point applications and their outgoing connections.

ServiceWatch comes with predefined discovery patterns for a large number of predefined applications (CI Types).

However, if you a define new CI type (for example, for an in-house application or an off-the-shelf application for which no CI types are currently defined), you must define discovery patterns for that CI type after completing the CI type definition.

A CI type can have multiple discovery patterns for any number of reasons. For example, the CI type might have or use:

- different protocols
- different operating systems
- different entry point types

Defining discovery patterns for a CI Type is essentially a two part process:

1. Defining a General Pattern Definition – this includes general descriptive information about the pattern, and also the list of Identification sections and Connectivity sections:
   - ✓ Identification Sections – an identification section identifies an entry point type that can point to the CI Type
   - ✓ Connectivity Sections – a connectivity section identifies a type of outgoing connection to which the particular CI type can connect (which, in turn, can be an entry point for an object lower down in the business service hierarchy).

The screen used for defining this information is fairly static and unchanging.

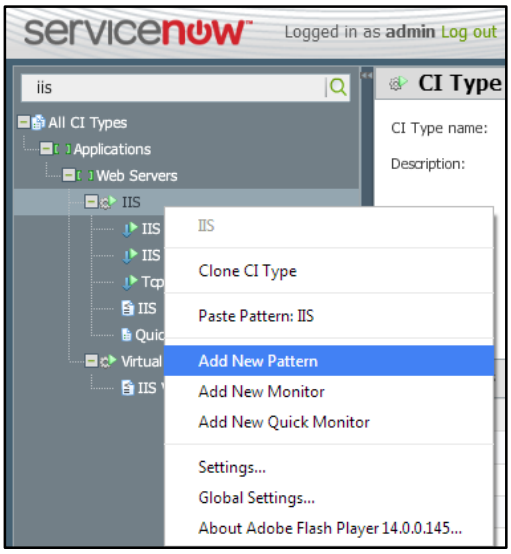
2. Chapter 5: Defining Discovery Step Details – For each identification section and connectivity section that you identified in the general pattern definition, you must define the detailed steps that discovery must perform to be successful.

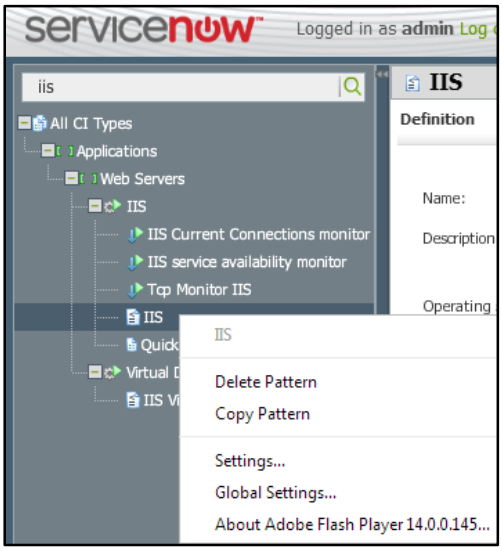
The screen used for defining these steps is quite dynamic.

Defining a General Pattern Definition

This procedure assumes that you are defining new discovery patterns for a CI Type. However, you can also edit existing patterns, and this will be noted in the procedure. An example that illustrates all of the steps for creating a new pattern is described in Appendix A: Apache on Unix pattern.

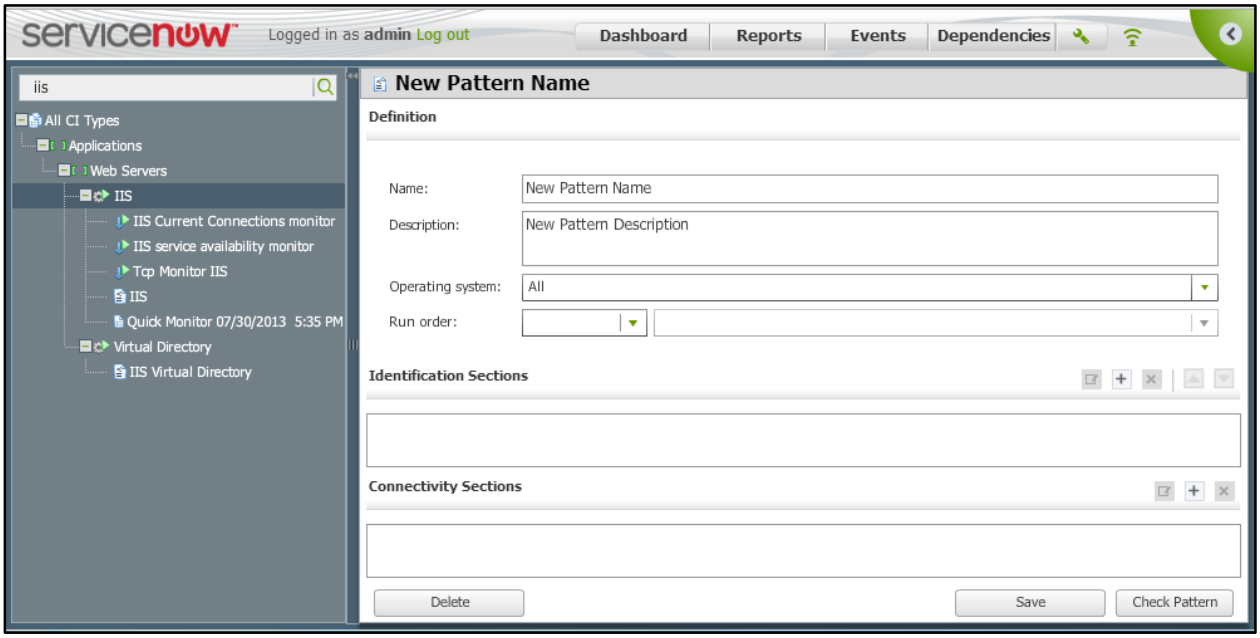
1. Click  in the Menu Bar. In the list of **Settings** options, click the **CI Types, Patterns & Monitors Definitions** link.
2. In the Configuration tree, display (and expand, if possible) the CI Type for which you want to define discovery patterns.
3. Add a new set of Discovery patterns for the CI type by right-clicking the CI type, and in the pop-up menu select **Add New Pattern**. Or, you can select and right-click an existing pattern, select the **Copy Pattern** option, and then modify its definition.



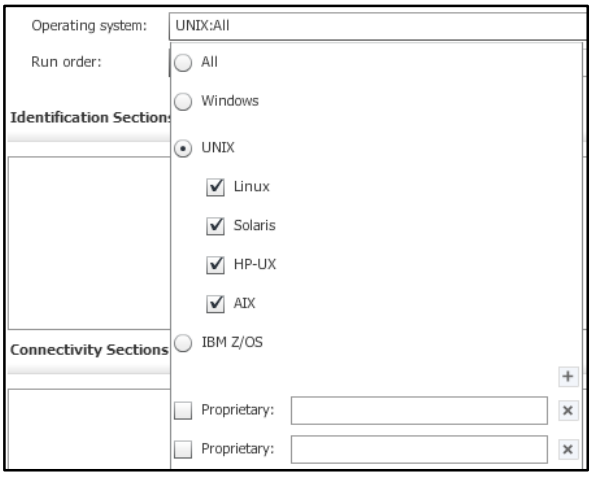


Note: To edit an existing pattern for a CI type, click that pattern in the configuration tree under the CI type. The basic discovery pattern screen is displayed (FIGURE 3). The screen title bar displays the pattern name character-by-character as the **Name** field is being filled in.

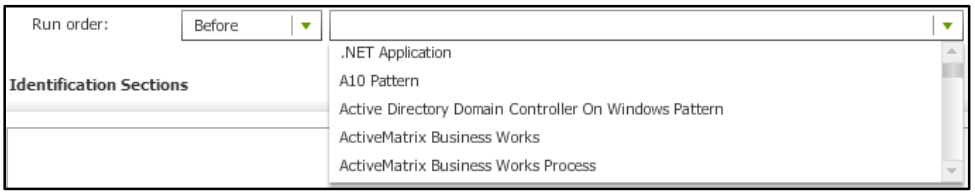
Figure 3: Pattern Name screen



4. Fill in the general details about the pattern in the top part of the screen as follows:
   a. Specify a **Name** (mandatory) and a **Description** (optional) for the pattern. This name will appear in the configuration tree.
   b. Select the **Operating system** from its drop-down list:



   c. If this pattern always runs before or after another pattern, select **Before** or **After** in the **Run order** field and then select the other pattern in the drop-down list.



Note: In most cases this field is not relevant. It is only relevant if a particular pattern can be confused with another pattern.

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, in the **Run Order** field in the MS Exchange pattern definition, select **Before** and **IIS**.

5. Fill in the **Identification Sections** as follows:
   a. To add an Identification section, click  in the **Identifications Sections** header.

Note: To edit an existing Identification section, select it and click  in the header.

b. Fill in the **New Identification Section** dialog box as follows:

i. Specify a **Name** for the Identification Section.

ii. In the **Entry point types** drop down list, select all relevant entry point types.

iii. In the **Find Process Strategy** drop down list, select the appropriate strategy for finding the process that will populate the **Process** variable in the **Temporary Variables** table. The appropriate strategy is usually determined by the type of port on the entry point. Valid values:

- ✓ LISTENING_PORT – Select this value if the entry point port is a listening port.
- ✓ TARGET_PORT_AND_IP – Select this value if the entry point port communicates with another server.
- ✓ NONE – Select this value if the Process variable should not be populated.

iv. Click the Create New button.

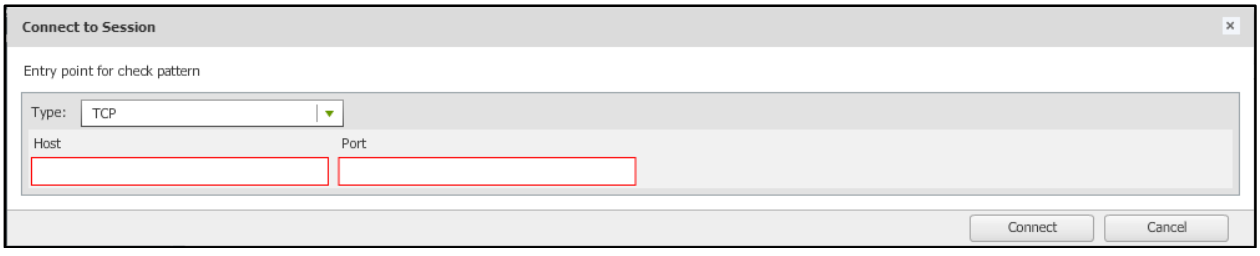
6. After you have created an **Identification section**, define its discovery pattern steps. For instructions, see Defining Discovery Steps – General Procedure.
7. Repeat steps 5 and 6 until you have defined all Identification Sections.
8. Fill in the **Connectivity Sections** as follows:

a. To add a Connectivity section, click + in the Connectivity Sections header.

Note: To edit an existing Connectivity section, select it and click the icon.

b. In the **New Connection Section** dialog box, specify a name for the Connectivity Section and then click the Create New button.

9. After you have created a Connectivity section, define its discovery pattern steps. For instructions, see Defining Discovery Steps – General Procedure.
10. Repeat steps 8 and 9 until you have defined all Connectivity Sections.
11. When you finish defining the pattern, click Save at the bottom of the screen.
12. To check if the defined pattern leads to a valid CI Type, click the Check Pattern button, fill in the **Connect to Session** dialog box, and click its Connect button.



Chapter 5: Defining Discovery Step Details

This chapter contains the following topics:

Concepts

For each identification section and connectivity section that you identified in the general pattern definition, you must define the detailed steps that discovery must perform to be successful. These steps constitute the "nuts and bolts" of discovery.

You can choose to define these steps immediately after adding an Identification or Connection section, or you can choose to define the steps after adding all Identification and Communication sections.

The screen you use for defining these steps is quite dynamic. It varies according to the operation being performed in any particular step, and will even vary further according to values selected during step definition.

Before beginning to define discovery steps, you should become familiar with the layout of the Discovery Step Definition screen and the operations that you can perform in this screen.

Working in the Discovery Steps Screen

This section contains the following topics:

Understanding the Screen Layout

The Discovery Steps pattern editor (Figure 5), or a variation of it, is displayed when you double-click an Identification or Connectivity Section in the **Pattern Name** screen.

Figure 4: Identification Section pattern whose discovery steps will be defined or modified

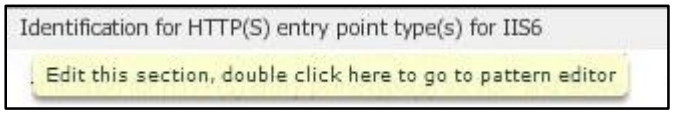
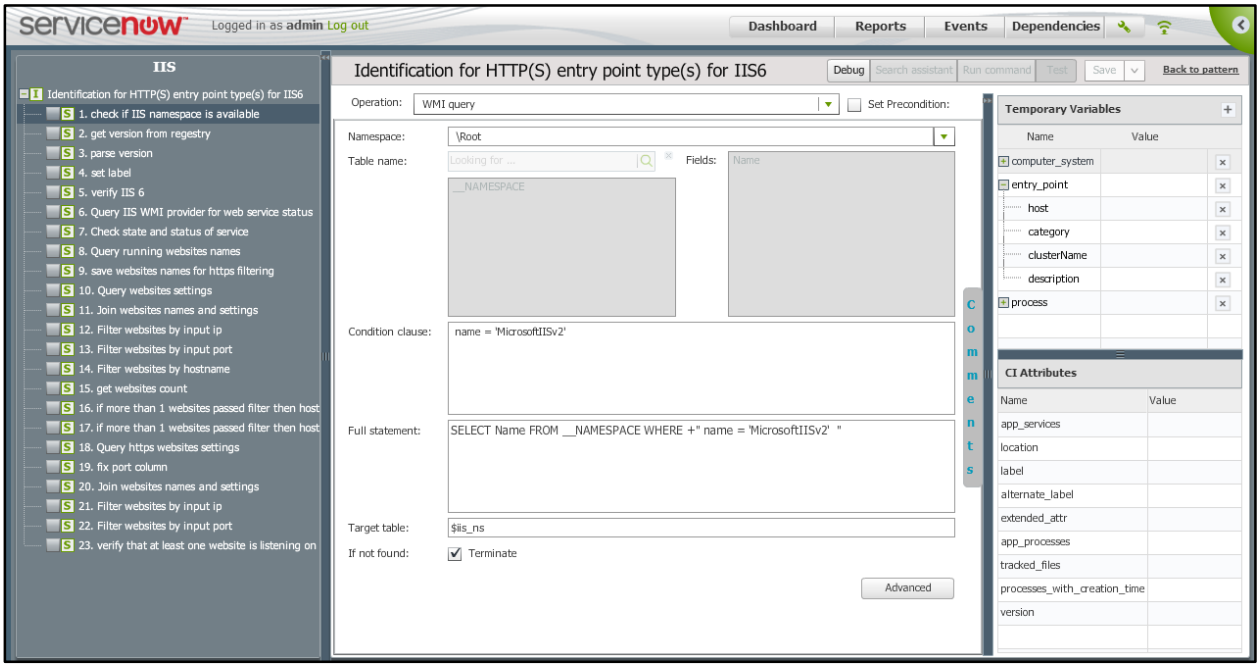


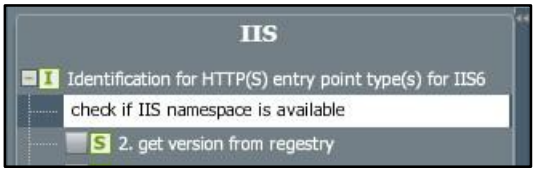
Figure 5: Discovery Steps pattern editor screen



This screen contains the following sections:

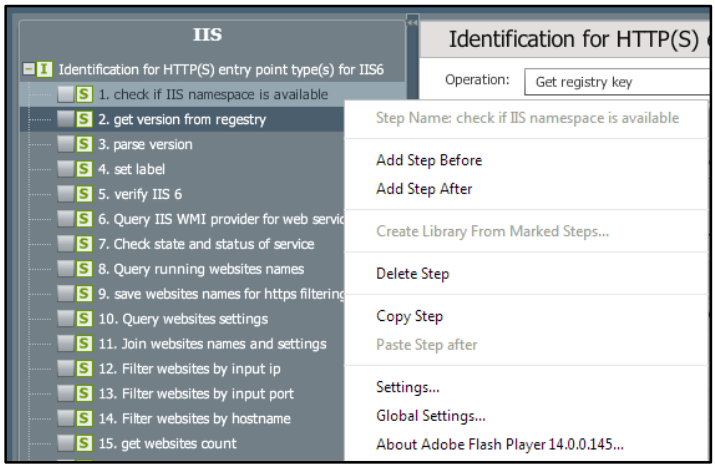
- **Operational Step Tree nodes** displayed in the left pane. Each node represents an operational discovery step. Clicking a tree node displays its operational steps.

  Double-clicking a node lets you rename the step by typing over the current step name.



Right-clicking a tree node displays a menu containing actions you can select.

Figure 6: Entry Point tree node Right-Click Menu



- **Operational Step Definition** – In the main center pane, you define (or view already defined) operational steps. When defining a step, first click the **Operation** down-arrow and select an operation that determines the fields that need to be displayed and filled in. The following Operations are described in Table 1 on page 29 and in detail by topic on pages 33 - 70:

Change User	Create Connection	Filter Table
Find Matching URL	Get Process	Get Registry Key
LDAP Query	Library reference	Match
Merge Table	Parse command output	Parse file
Parse URL	Parse variable	Put File
Set Variable	SNMP Query	Transform Table
Unchange User	Union Table	WMI Method Invocation
WMI Query		

- Two tables are displayed on the right side of the screen:
  - ✓ **Temporary Variables** – In the upper right pane, this table displays temporary variables generated by previous steps in the discovery pattern.
  - ✓ **CI Attributes** – In the lower right pane, this table displays CI attributes and values defined in the CI Type Definition screen.

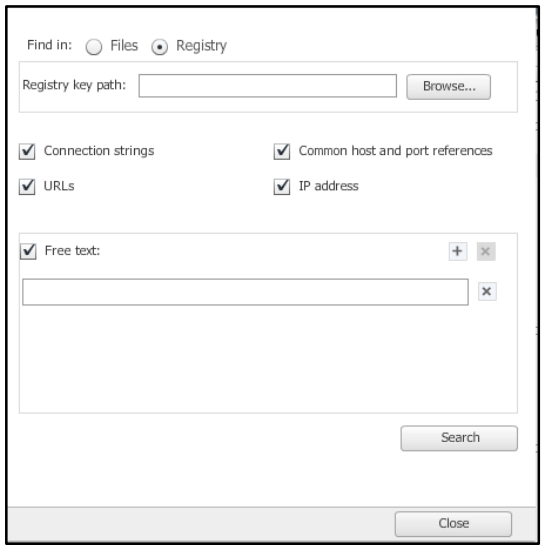
  You can add variables to the **Temporary Variables** table by clicking the + in the table header.

  You can remove or disable an existing variable by clicking the × on its row.

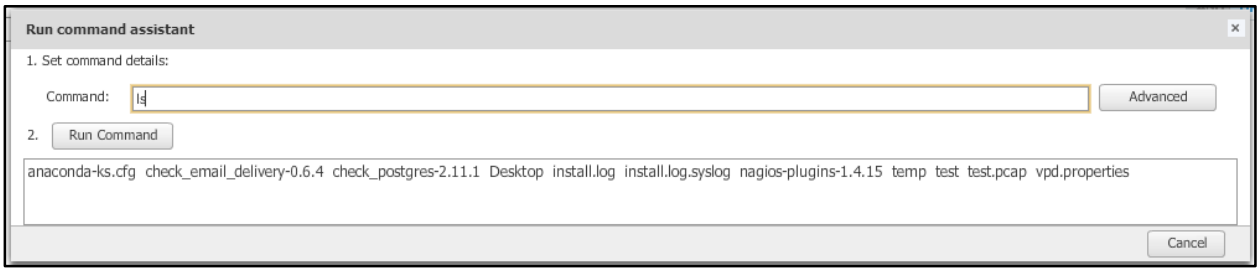
- **Action Buttons** and **Links** are displayed at the top of the screen. The following actions are possible:
  - ✓ **Debug** – In Debug mode, the editor interactively performs all actions that you perform in the screen. It is recommended that you work in Debug mode. When Debug mode is active, a

green check mark appears in the Debug button. For instructions on activating Debug mode, see ACTIVATING DEBUG MODE.

✓ **Search assistant** – Helps find Connection strings, Common host & port references, URLs and IP addresses of files or registry entries for patterns and elements on the target machine. It only works in Debug mode. Clicking this button displays the Files **Search assistant** dialog box. Clicking the **Registry** radio button displays the Registry **Search assistant** dialog box.



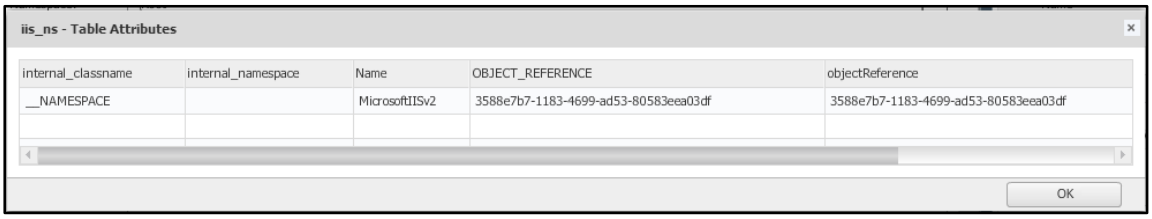
✓ **Run command** – Clicking this button displays the **Run command assistant** dialog box. It only works in Debug mode. For example:



Clicking Advanced toggles between displaying and not displaying these additional fields:

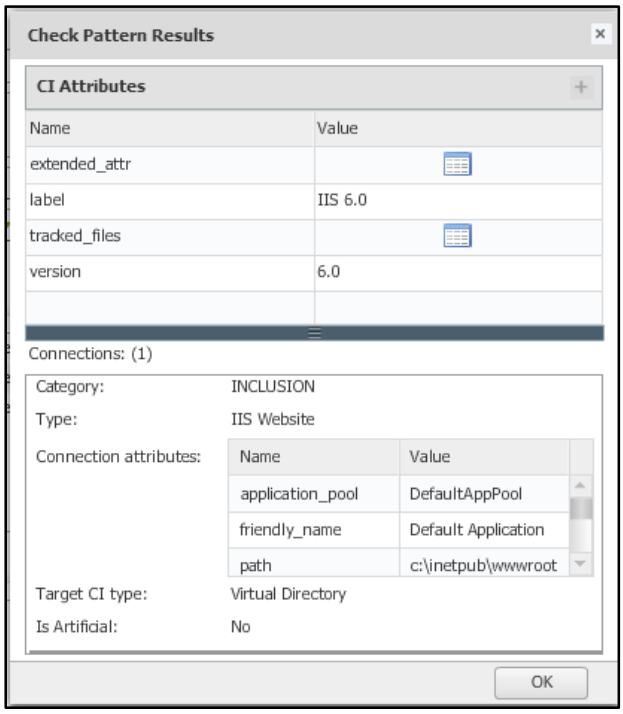
Clicking Run Command runs all the steps on the target machine to see if they work, checks that all required attributes have values, that no errors occurred during execution, and that discovery eventually leads to a valid CI.

- ✓ **Test** – Tests the current step if Debug mode is active. The **Table Attributes** list will be displayed. For example:



Note: When you finish defining a step and ask to create a new step, the step you just finished is automatically tested without the Test button being clicked.

- ✓ **Check Pattern** – After the discovery pattern is completely defined, click Check Pattern to determine if the patterns works properly. A Check Pattern Results box is displayed:



- ✓ **Back to pattern** – Clicking this link redisplays the **Pattern Name** screen.

Caution: Do ***not*** click the browser back button. Doing so will take you out of the ServiceWatch application.

Working with Variables

Variables play a big role in discovery pattern definition.

In the discovery step definition screens, variables are generally displayed in two locations:

- **Temporary Variables** table
- **CI Attributes** table

There are several types of variables:

- Scalar variables are the most common type of variable. They hold a single value (a single string).

- Tabular variables behave as if they are a table (matrix). They have rows and columns. Rows are identified by row number. Columns are identified by name.
  Note: Each cell in a tabular variable is itself a scalar variable.
  In the **Temporary Variables** table, tabular variables are identified by a table icon next to the name. If you click the icon, it displays the table that constitutes the variable.
- Vector variables consist of a single, unnamed column with as many rows as needed.
  Note: This unnamed column is in contrast to the named columns in a tabular variable.
- Special "Container" variables can hold any combination of scalar and tabular variables.
  Only three types of "container" variables exist:
  - ✓ Computer System – contains information about the host to which you are connected
  - ✓ Entry Point – contains information about the entry point
  - ✓ Process – contains data about the process detected at the port pointed to by the entry point

  "Container" variables only appear in the **Temporary Variables** table, not the **CI Attributes** table. They are indicated by a folder icon which, when clicked, displays the contents of the container.

Variables are always prefixed by a **$** to indicate they are variables. However, the **$** 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** and it will appear in the **Temporary Variable** table as Abc.

Activating Debug Mode

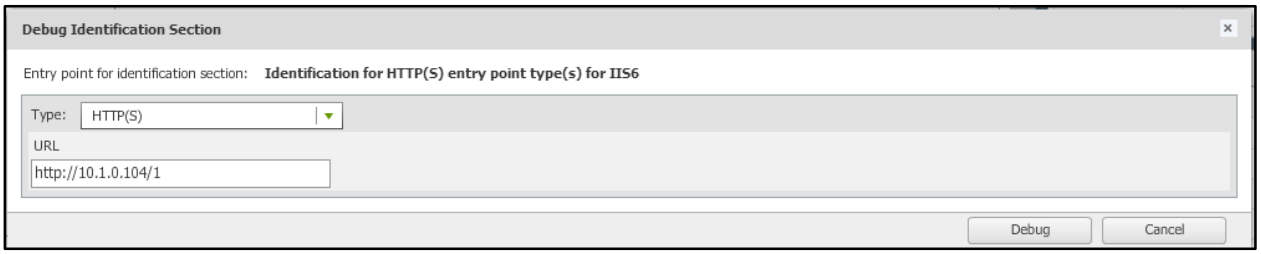
It is recommended that you work in interactive Debug mode. In Debug mode, the editor actually performs all actions that you perform in the screen. For example, in Debug mode:

- Browse and retrieve actions will really work and tests will really run.
- The **CI Attribute** and **Temporary Variable** tables will display values.

If you are not in Debug mode, actions are not performed and values are not displayed in the tables. When Debug mode is active, a **green** check mark appears in the Debug button.

If no green check mark appears, do the following to activate Debug mode:

1. Click **Debug**. The **Debug Identification Section** *or* **Connection Section** dialog box is displayed.



2. Fill in the required details for the Entry Point type.
3. In the dialog box, click **Debug**. The dialog box closes, Debug mode is activated, and a **green** check appears in the **Debug** button in the screen: 

.
4. To deactivate Debug, click the **Debug** button in the screen. Debug mode is deactivated and the **green** check disappears.

Alternative Method for Defining Many Operations – in Debug Mode

Working in **Debug** mode provides an alternative method for performing many operations.

Many operations require that you specify a specific value (or identifying information) from a specific source. For example, you might need to specify a particular value, or a particular location and delimiter, in a particular file. If you are *not* in **Debug** mode, you need to remember and 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. For operations requiring such information, this is a much easier process.

Operations where Debug can be useful usually contain buttons/fields for using browse and select functions in addition to fields for manually defining details. If you are not in **Debug** mode, you must manually define the details.

Dragging & Dropping and AutoCompletion

When filling in fields with values, you can use the Drag-and-Drop and AutoCompletion features:

- **Drag-and-Drop** –Select a value in a table or field and drag-and-drop it into another field.
  Example
  Suppose the **CI Attributes** table (in the right pane) contains the value **$process.portsList** and you want to place this value in a field in the center pane. Instead of typing the value, you can select the value in the table and drag it to the target field.
- **AutoCompletion** – Variables and several other objects have a **$** prefix. When you begin typing characters with a $ prefix into a field, a drop-down list shows all currently available values that satisfy the currently entered characters. As you type more characters, the drop down list shortens. If only one choice remains, that value is automatically entered into the field.
  Example
  Typing **$P** in a field will display a list of possible values beginning with **$P**. You can then choose the correct value.

Concatenating Variables and Constants

You can specify complex (concatenated) values in fields.

To concatenate values in a field, specify a + sign between the values. When concatenating values in a field, keep in mind the following:

- As mentioned in DRAGGING & DROPPING AND AUTOCOMPLETION, variables begin with **$** and you can use the drag-and- drop feature or AutoCompletion to fill in fields.
- You can indicate constants by enclosing them in quotes (e.g., "**uuid=**").

The value **`"uuid="+$entry_point.uuid`** consists of a constant concatenated with a variable.

Defining Conditional Criteria

For many operations, you can define conditional criteria. For example, you can define prerequisite conditions, filtering criteria, matching criteria. The format is generally the same – two fields separated by an operator.

To define conditional criteria, do the following:

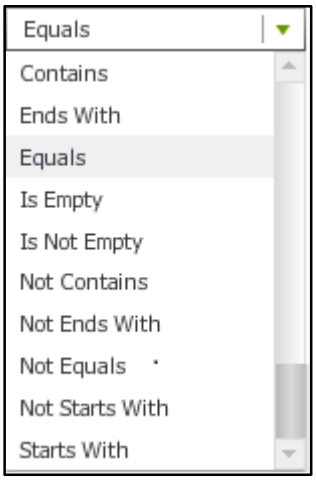
1. Define a condition as follows:

   Note: You can use the Drag-and-Drop, AutoCompletion and Concatenation when filling in fields. See DRAGGING & DROPPING AND AUTOCOMPLETION and CONCATENATING VARIABLES AND CONSTANTS.

   a. In the first field in the condition, specify the name of the item (e.g., a variable name).

   b. Select an operator from the drop-down list. The default value is **Contains**. If you select **Is Empty**, the second field is irrelevant and disappears. The entire operator drop-down list is illustrated here:

Figure 7: Operator Drop-Down List



   c. In the second field, specify a value for the item specified in the first field.

To add conditions, perform the following steps:

1. Click the + icon.

   Note: If multiple conditions are needed, instead of adding multiple conditions, consider defining multiple steps, each with only one condition.

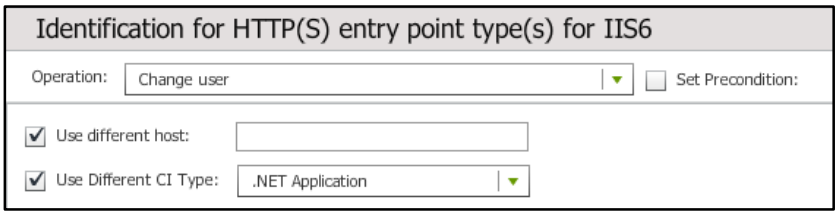
2. If you define multiple conditions, set the logical relationship between them by clicking the AND or the OR half of the AND OR icon.

3. To remove a condition, click the × on that condition line.

Figure 8: Multiple Condition logical operator

Making Performance of a Step Conditional

The **Operation** drop-down list on the first line in the **Discovery Steps Definition** screen has a **Set Precondition** checkbox.

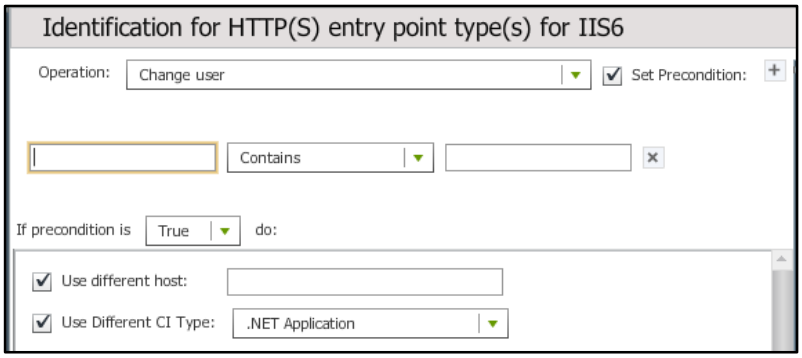


By default, this checkbox is *not* checked and performance of the step is unconditional; that is, the step is always performed as defined.

To make performance of a step conditional on specified criteria being satisfied (or unsatisfied):

1. Select the **Set Precondition** checkbox. The display expands to enable you to specify criteria including a condition.

Figure 9: Set Precondition checkbox



2. Fill in the condition. To add more conditions, click the + icon. For information about filling in conditions, see DEFINING CONDITIONAL CRITERIA.
3. Use the **If precondition is** drop-down list to indicate whether step performance depends on the criteria (preconditions) being satisfied (**True**) or not satisfied (**False**).
4. To make a conditional step unconditional, uncheck (clear) the **Set Precondition** checkbox.

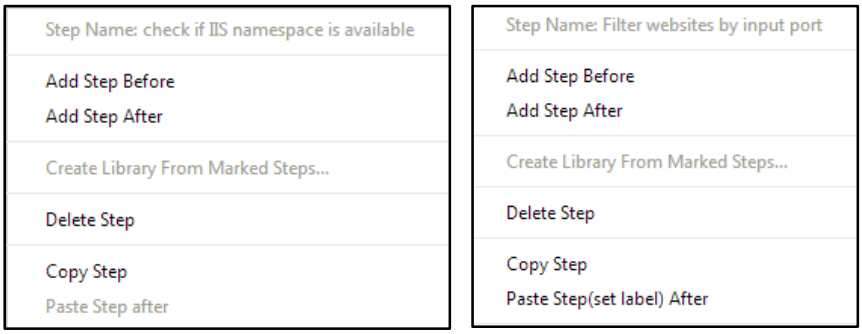
Manipulating Steps

For each step you define, a tree node representing the step is displayed in the left pane.

Notes:

- When a step is first displayed, its default name is **_Untitled_Step_**. While *not* in Debug mode, double-click that tree node default name and overwrite **it** with a meaningful name. Use this same procedure to change the name of any existing step.
- To display the details of any step, click that step's tree node.
- Placing the cursor anywhere over a step's tree node displays its full step name in a tool tip.

- Right-click any step's tree node to display its right-click menu. This menu enables you to add, delete, and copy a step. After a step has been copied, the **Paste Step(<step name>) After** option is operational.



- To re-sequence already defined steps, use the **Copy Step**, **Paste Step(<step name>) After** and **Delete Step** options in the tree's right-click menu.

Manipulating Tables

ServiceWatch automatically populates the **Temporary Variables** and **CI Attributes** tables with item names and, if **Debug** mode is active, with their values. However, you can also add temporary variables.

To display values in the tables if they are not displayed, activate Debug mode. For instructions, see ACTIVATING DEBUG MODE.

To add a temporary variable:

1. Click the + icon in the **Temporary Variables** header and type the variable's name in the activated text box in the **Name** column.
2. To remove any temporary variable, select it and click its × icon.

Defining Discovery Steps – General Procedure

Notes:

- Before defining a discovery pattern, plan the logical flow of steps required by the discovery pattern, including what each step should accomplish. Later, you will select the appropriate operations based on the planned flow.
- To display the Discovery Steps pattern editor screen **(**Figure 5 on page 18) for an Identification section or Connectivity section in the Pattern Name screen (Figure 3 on page 14), double-click a row in the desired section.
- To return to the **Pattern Name** screen from the Discovery Steps pattern editor screen, click the **Back to Pattern** link.

To define discovery steps for a particular Identification section or Connectivity section in a CI type pattern:

1. With the pattern for the CI type displayed in the **Pattern Name** screen, select the desired Identification section or Connectivity section entry and double-click it. Its Discovery Steps pattern editor screen is displayed.

   If no Discovery steps have been defined for this pattern, an **_Untitled_Step_** tree node is displayed and automatically selected at the left. Otherwise, the tree node of the first defined step is automatically selected.
2. To define or edit the details of a step that already has a step tree node, select the desired step node if it is not already selected.
   a. In its **Operation** drop-down list, choose the desired operation.
   b. Fill in the details of the operation.
   c. Assign a meaningful name to replace the default name **_Untitled_Step_**) as follows:
      i. If in Debug mode, exit Debug mode.
      ii. Double-click the tree node with the default name **_Untitled_Step_**.
      iii. Overwrite the default step name with a meaningful name.
      iv. Click the tree root or any other node.
3. To add new a step:
   a. Right-click the step that is immediately before the step you need to add and select the **Add Step After** option, *or*
   right-click the step that is immediately after the step you need to add and select the **Add Step Before** option.
   b. Define the step details as described in step 2 above.
4. It is recommended that you activate **Debug** mode, if it is not already active. (A green check mark in the Debug button indicates that Debug mode is active.) For information about activating Debug mode, see ACTIVATING DEBUG MODE.
5. In the main (center) section of screen, select the appropriate operation for the step. For information that can help you choose the appropriate option, see SELECTING THE APPROPRIATE OPERATION FOR A STEP. The fields that are displayed in this section of the window vary according to the selected operation.
6. By default, the step will be performed unconditionally. To make performance of the step conditional upon certain criteria being satisfied or not satisfied, select the **Set Preconditions** check box and define the criteria. For more information on defining criteria, see MAKING PERFORMANCE OF A STEP CONDITIONAL.
7. Define the details of the step. Details vary according to the operation type. Instructions for filling in details for each operation type are provided, by operation type, in DEFINING STEP DETAILS FOR THE SELECTED OPERATION.
   TABLE 1 on page 29 provides links to the instructions by operation type.

   When defining details, you can utilize the procedure and features defined under WORKING IN THE DISCOVERY STEPS SCREEN. For example, you can use the DEFINING CONDITIONAL CRITERIA procedure and the DRAGGING & DROPPING AND AUTOCOMPLETION feature.

8. When you finish defining the details of a discovery step, you can create the next discovery step by repeating steps 3 through 7.

   Note: After completing a step, when you request creation of a new step, a test is automatically run on the completed step if you are in Debug mode.

9. When you finish defining all the steps for the section, you can test the entire set of steps by clicking **Check Pattern**. For **Check Pattern** to be effective, Debug mode must be active.
10. When you satisfied with the definition, click **Save**.
11. To return to the CI type's pattern definition in the General Pattern Definition screen, and the Discovery Pattern configuration tree containing CI types and their patterns, click the **Back to pattern** link.

    Caution: Be sure to click **Save** before clicking **Back to pattern**, or the changes will be lost.

Selecting the Appropriate Operation for a Step

When you know what a step should accomplish, use TABLE 1 below to select the appropriate operation. This table lists objectives that steps can accomplish and the operation appropriate to each objective.

Notes:

- For a list of the step operations used in several real discovery patterns, see EXAMPLES OF STEPS AND OPERATIONS DEFINED FOR A PATTERN SECTION.
- CHAPTER 5: DEFINING DISCOVERY STEP DETAILS provides detailed instructions for each operation.

Each operation in TABLE 1 contains a link to the instructions for performing that operation.

Table 1: Discovery Step Operations

To accomplish the following objective	Perform this operation
Extract information from the output of a command	Parse Command Output. See PARSE OPERATIONS
Extract information from a file	Parse File. See PARSE OPERATIONS
Extract information from a variable	Parse Variable. See PARSE OPERATIONS
Execute a WMI query	WMI QUERY
Execute an SNMP query	SNMP QUERY
Terminate if a condition is not met	MATCH
Break down a URL into its components	PARSE URL
Provide information about outgoing connections	CREATE CONNECTION
Query for registry keys	GET REGISTRY KEY
Filter a table according to specified criteria	FILTER TABLE
Merge two tables	MERGE TABLE
Add computed columns to an existing table	TRANSFORM TABLE
Set a variable's value	SET VARIABLE
Transfer a file to the remote computer	PUT FILE
Search for processes according to specified criteria	GET PROCESS
Find the best match for a URL in a list of URLs	FIND MATCHING URL
Append two tables that share the same format	UNION TABLE
Use applicative credentials instead of the default administrative credentials	CHANGE USER
Switch back to the default administrative credentials	UNCHANGE USER
Query an LDAP directory	LDAP QUERY
Execute a method using WMI (Windows Management Instrumentation)	WMI METHOD INVOCATION
Combine a number of steps to be executed as a group	Library reference

Examples of Steps and Operations Defined for a Pattern Section

This section provides examples that illustrate what operational discovery steps might be defined for a particular Identification section or Connectivity section.

Each table in this section deals with a pattern for a particular CI type, lists the names (but not the details) of the steps, and indicates their respective operations.

This section contains the following examples:

- SUN JES PATTERN – IDENTIFICATION FOR SMTP ENTRY POINT
- SUN JES PATTERN – MSS CONNECTION
- J2EE EAR ON WINDOWS – EAR TO MQ CONNECTIVITY

Sun JES Pattern – Identification for SMTP entry point

Hierarchy	Applications > Mail Services
CI Type	Sun JES
Pattern	Sun JES pattern
Section	Identification for SMTP entry point

STEP NUMBER AND NAME		**OPERATION**
1	Check process name to match SUNWmsgsr	Match
2	Calculate install_path	Parse Variable
3	Calculate Storage	Parse Command Output

Sun JES Pattern – MSS Connection

Hierarchy	Applications > Mail Services
CI Type	Sun JES
Pattern	Sun JES pattern
Section	MSS connection

STEP NUMBER AND NAME		**OPERATION**
1	Terminate if this is MSS	Match
2	LDAP base dn	Parse Command Output
3	Find mss connection through LDAP	LDAP Query
4	Create mss connection	Create Connection

J2EE EAR on Windows – EAR TO MQ Connectivity

Hierarchy	Applications -> Application Servers
CI Type	Websphere EAR
Pattern	J2EE EAR On Windows
Section	EAR To MQ Connectivity

STEP NUMBER AND NAME		**OPERATION**
1	Get queue jndi names from ibm-web-bnd.xml	Parse File
2	Get queue jndi names from ibm-application-client-bnd.xmi	Parse File
3	Get queue jndi names from ibm-ejb-jar-bnd.xmi	Parse File
4	Union queue_jndi_names1 with queue_jndi_names2	Union Table
5	Union queue_jndi_names1 with queue_jndi_names3	Union Table
6	Calc base_was_dir location	Parse File
7	Get q factories	Parse File
8	Merge the q factories with the jndi ref names	Merge Table
9	Create connection to MQ	Create Connection

Defining Step Details for the Selected Operation

This section contains a subsection for each operation type. Each subsection explains how to fill in the step details for that operation type. For an explanation of which operation is appropriate for what you want to accomplish, see SELECTING THE APPROPRIATE OPERATION FOR A STEP on page 29.

Notes

- Fields displayed in the Discovery Step Definition screen vary according to the selected operation and may also vary based on values specified in the step definition. However, the type of section (Identification section or Connectivity section) does **not** impact what fields are displayed.
- Some operations contain Browse or similar buttons. These buttons provide an alternative to manually filling-in fields. For example, when specifying filtering criteria, instead of trying to remember the correct values and type them correctly, the operation might let you browse to a file and populate the filter fields with the correct data. In these cases, you are not obligated to browse (you can fill in the fields manually), but browsing makes the task easier.
- When you request an operation, the step's details are usually blank (except for default values). However, in some cases, it is useful to see how a screen might look after it is filled in. Therefore, screens in this section often show completed step definitions that have been taken from various CI types and patterns. They do not represent all of the screens for a single pattern definition.
- For each operation type, you can make performance of the step conditional. For more information, see MAKING PERFORMANCE OF A STEP CONDITIONAL on page 25.

This section explains how to fill in step details for the following operations:

- CHANGE USER
- CREATE CONNECTION
- FILTER TABLE
- Find Matching URL
- GET PROCESS
- GET REGISTRY KEY
- LDAP QUERY
- LIBRARY REFERENCE
- MATCH
- MERGE TABLE
- Parse Command Output. See PARSE OPERATIONS
- Parse File. See PARSE OPERATIONS
- Parse Variable. See PARSE OPERATIONS
- PARSE URL
- PUT FILE
- SET VARIABLE
- SNMP QUERY
- TRANSFORM TABLE
- UNCHANGE USER
- UNION TABLE
- WMI METHOD INVOCATION
- WMI QUERY

Change User

Description

Purpose Use application credentials instead of the default administrative level credentials.

By default, discovery uses OS Admin credentials. However, these credentials might not be sufficient. For example, an Oracle pattern running under the OS needs Oracle credentials; MSExchange and MQ require special credentials.

Defining a Change User operation causes the discovery process to seek the appropriate credentials for the same or a different CI type on the same or a different host.

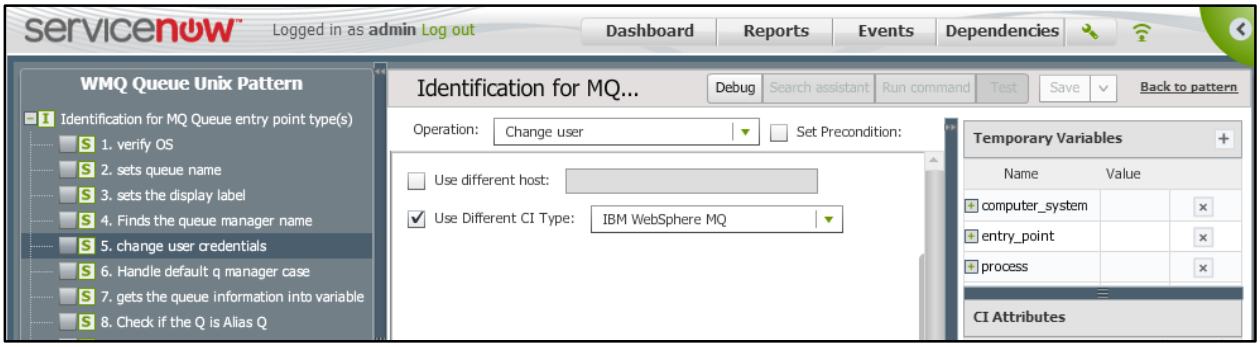
Instructions

1. To find credentials on a different host, select the **Use different host** checkbox and fill in the host name.
2. To find credentials for a different CI type, select the **Use Different CI Type** checkbox and select the CI Type from the drop-down list.

Note: To return to using default administrative level credentials at a later point in the discovery process, insert a step and select the UNCHANGE USER operation.

Example

Figure 10: Change User Credentials



Example taken from:	
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern
Section	Identification for MQ Queue entry point type(s)
Step # and Name	5. Change user credentials

Create Connection

Note: This operation is available only for Connectivity sections.

Description

Purpose Provide information about an outgoing connection.

Instructions

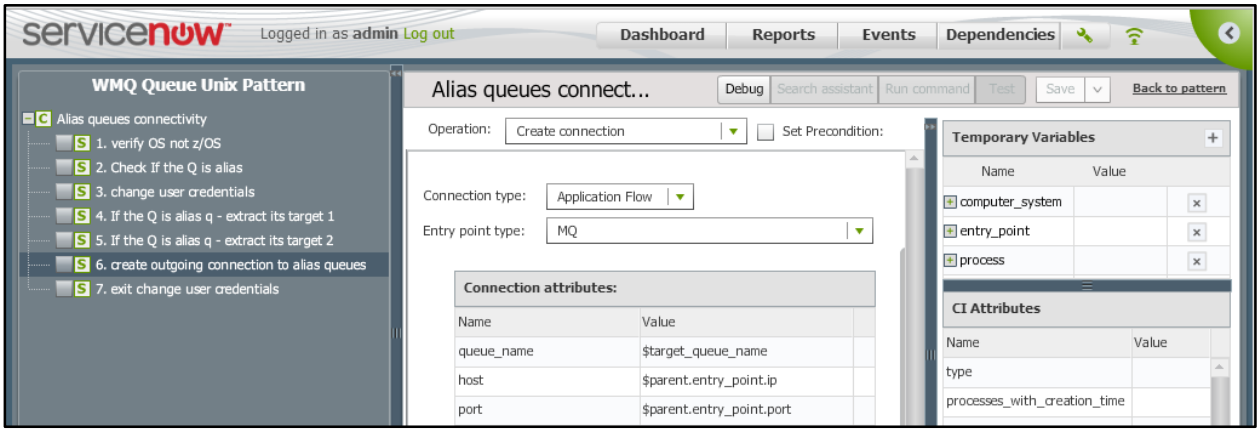
1. Select the **Connection type** (**Application flow**, **Storage flow**, **Cluster**, or **Inclusion**) from the drop-down list.

   Note: Inclusion means that you are connecting to an object that is included in the current object. Examples of Inclusion connection types are: a connection from J2EE to EAR, and a connection from IIS to Website.

2. If you selected **Cluster** as the connection type, a **Cluster Name** field is displayed. Fill in the cluster name.
3. Select the **Entry point type**. The **Connection attributes** change accordingly.
4. Fill in the **Connection attributes**.
5. Select a **Target CI type** from the drop-down list.
6. If the connection should be hidden (that is, not shown in the user interface) but is used for continuing the discovery flow, select the **Is Artificial** checkbox.

Example

Figure 11: Connection attributes table



Example taken from:	
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern
Section	Alias queues connectivity
Step # and Name	6. Create outgoing connection to alias queues

Filter Table

Description

Purpose Filter a table according to specified criteria.

This operation checks a selected source table for specified field-value criteria. If they are found, it places those fields/values into a specified target table.

Notes:

- If the specified target table is already populated, the new fields/values from the source table overwrite the target table's previously existing fields/values.
- You can use the Drag-and-Drop and AutoCompletion features to copy table names and fields/values from the **Temporary Variables** and **CI Attributes** tables.

Instructions

1. Specify the **Source table**. **Hint**: You can use Drag-and-Drop or AutoCompletion.
2. Specify the **Target table**.
3. Specify the criteria for determining which values from the Source table are placed in the Target table. For instructions about specifying the criteria, see DEFINING CONDITIONAL CRITERIA.

Example

Figure 12: Specifying Source & Target Tables and criteria for moving values from source to target

Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	Websphere EAR
Pattern	J2EE EAR On Linux
Section	DB2 JDBC connectivity
Step # and Name	8. Filter DB2 JDBCProviders

Find Matching URL

Description

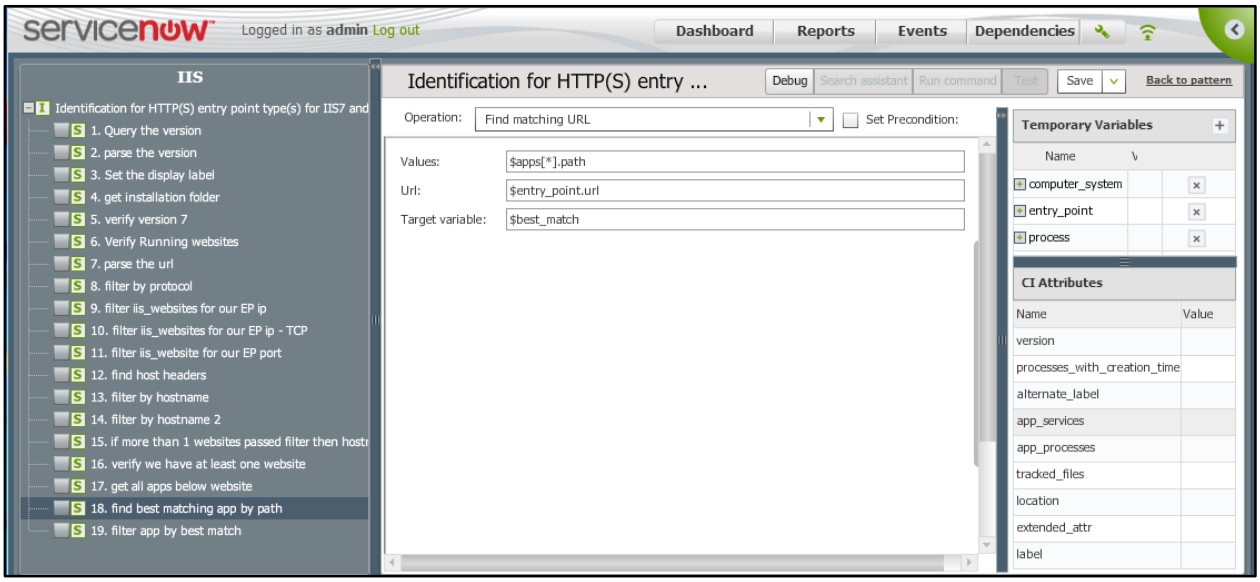
Purpose Find the best match for a URL in a list of URLs.

Instructions

1. In the **Values** field, specify the set of potential matching URLs.
2. In the **Url** field, specify the URL for which we are seeking the best match.
3. In the **Target variable** field, specify the variable in which the best match should be placed.

Example

Figure 13: Finding the best match between a URL and a list of potential matching URLs



Get Process

Description

Purpose Search for processes according to specified criteria.

This operation finds processes that satisfy specified filtering criteria and places the results in a specified tabular variable.

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

Processes that satisfy the specified filtering criteria are placed in a tabular variable whose name you assign. This tabular variable will appear in the Temporary Variables table.

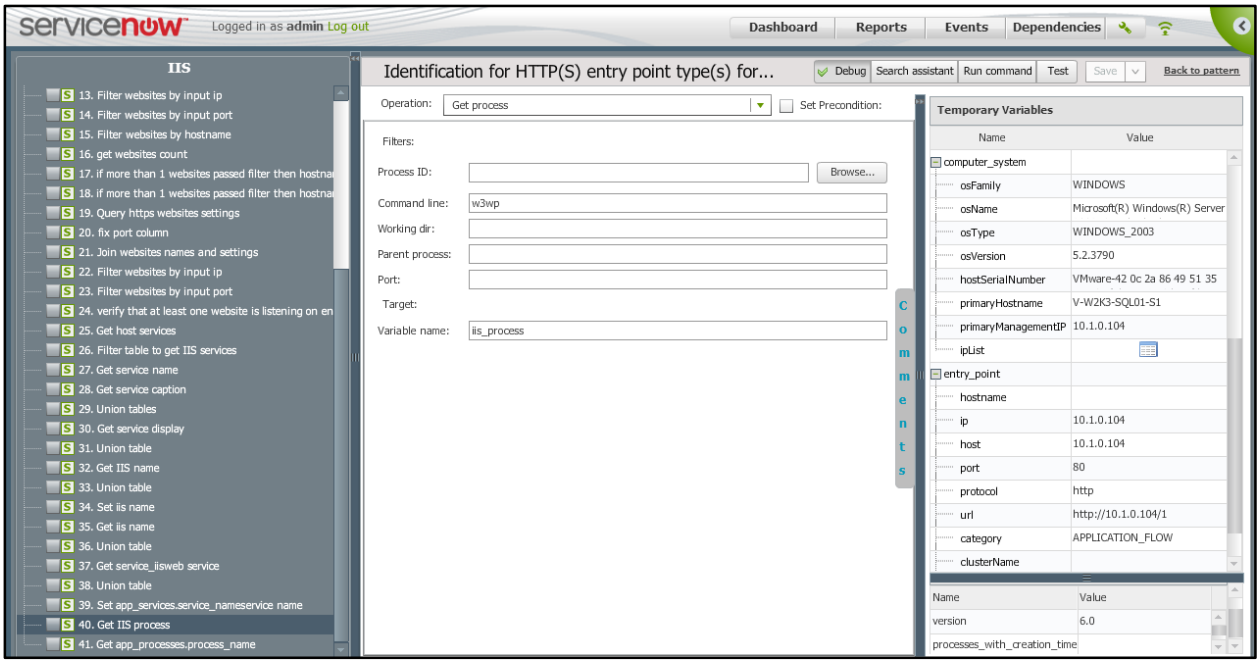
Instructions

1. Specify the filtering criteria in either of the following ways:
   - Manually fill in relevant filter fields.
   - Populate filtering fields with values from a selected process, as follows:
   i. Click the 

button to display a dialog box containing a list of processes on the machine.
   ii. Select a process and click **OK**. The dialog box closes and the filtering criteria are populated with values from the selected process.
   iii. Adjust the values in the fields as needed.
2. In the **Target: Variable name** field, specify a name for the tabular variable that will hold the list of processes that satisfy the filtering criteria. This variable will appear in the **Temporary Variables** list.

Example

Figure 14:Finding processes that satisfy specified filtering criteria



Example taken from:	
Hierarchy	Applications > Business Integration Software
CI Type	Web Servers - IIS
Pattern	IIS
Section	Identification for HTTP(S) entry point type)(s) for IIS6 second logic
Step # and Name	40. Get IIS process

Get Registry Key

Description

Purpose Query for registry keys.

This operation allows you to retrieve and select attributes from a registry key and store them in a table (tabular variable). For each selected attribute, you specify the name of a variable that will hold the attribute in the table.

You can specify the registry directory path and desired keys manually, or you can browse to the registry path and select the desired keys. When browsing you do not need to remember the exact directory path and key names and browsing avoids typing errors, but you must be in **Debug** mode to browse.

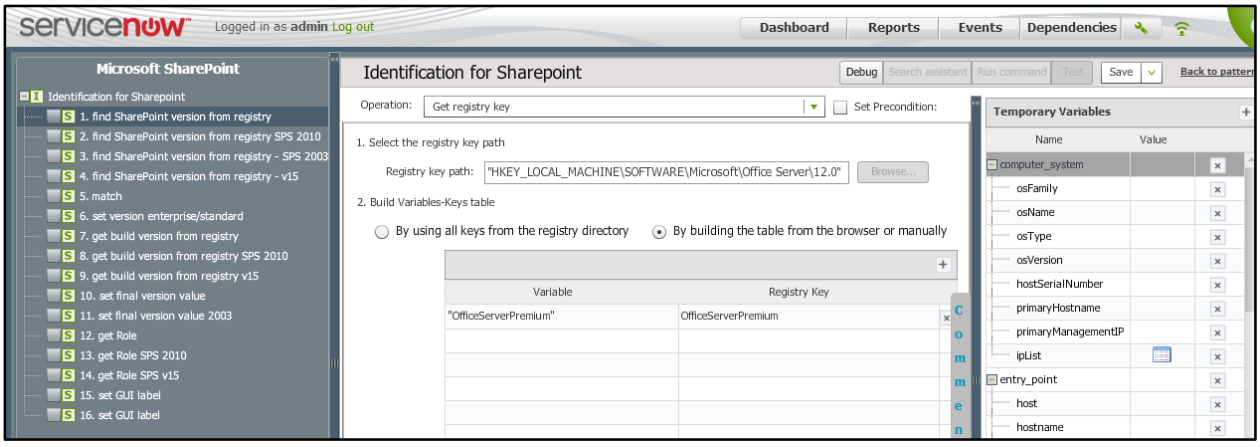
Instructions

Do either of the following:

- To manually define the registry query:
  1. In the **Registry key path** field, specify the Registry key path.
  2. In the Build variables-keys table:
     a. For each attribute that you want used for the query
        i. Click the  icon.
        ii. In the **Registry Key** column of the Variable—Keys table, specify the name of the attribute.
     b. In the **Variable** column, specify a name for the variable that will hold the value of the key in the Results table.
  3. If discovery should terminate if no results are found for the operation, select the **Terminate** checkbox.
- If you are in **Debug** mode, you can use the browse function to define the registry query.
  1. Click the **Browse** button and browse to and select the Registry key. The selected key path is placed into the **Registry key path** field. A dialog box opens and displays the list of keys in a tree on the left.
  2. Select the key. The right side of the dialog box displays the attributes for the key you selected.
  3. Select the attributes. They are placed into the **Registry Key** column of the **Variable—Keys** table.
  4. In the **Variable** column of the table, specify a name for the variable that will hold the value of the key in the Results table.
  5. If discovery should terminate if no results are found for the operation, select the **Terminate** checkbox.

Example

Figure 15: Obtaining a registry key



Example taken from:	
Hierarchy	Applications > Portals
CI Type	SharePoint
Pattern	Microsoft SharePoint
Section	Identification for Sharepoint
Step # and Name	1. find SharePoint version from registry

LDAP Query

Description

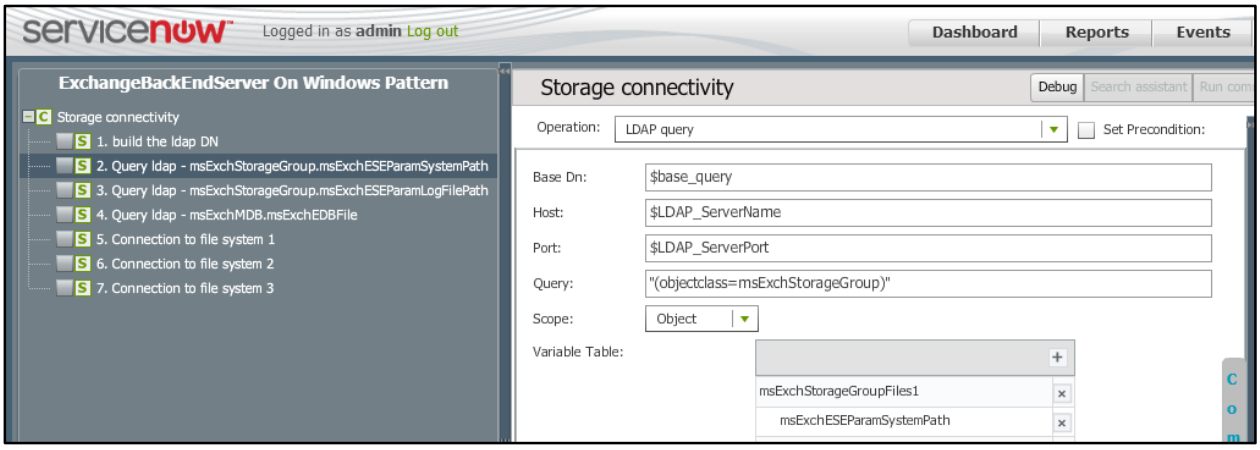
Purpose Query an LDAP directory.

Instructions

1. Fill in the **Base Dn**, **Host** and **Port** parameters.
2. In the **Query** field, specify the query.
3. From the **Scope** drop-down list, select one of these values:
   - ✓ Sub Tree – Look at the object and all of its descendants
   - ✓ Object – Look only at the object
   - ✓ One Level – Look at the object and one level below
4. In the **Variable Table**, define where to store the results:
   a. Click the Add button.
   b. Specify a table name and column name to hold the result of the query.
   c. For multiple results, click the Add button for each additional column needed and specify the column name.

Example

Figure 16: Querying an LDAP directory



Example taken from:	
Hierarchy	Applications > Mail Services
CI Type	ExchangeBackEndServer
Pattern	ExchangeBackEndServer On Windows Pattern
Section	Storage connectivity
Step # and Name	2. Query ldap - msExchStorageGroup.msExchESEParamSystemPath

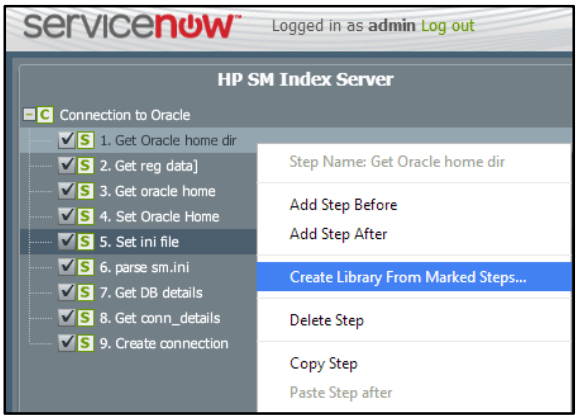
Library reference

Description

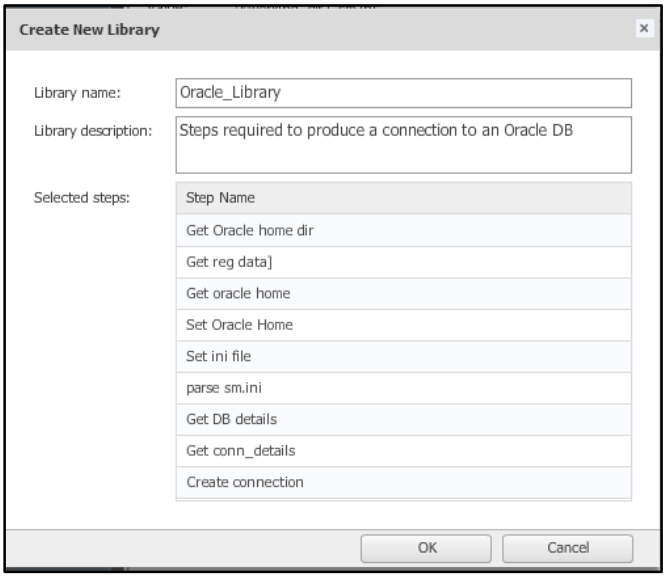
Purpose Combines a sequence of discovery steps, gives the sequence a unique name, and enables the sequence to be invoked as if it were a single step.

Instructions

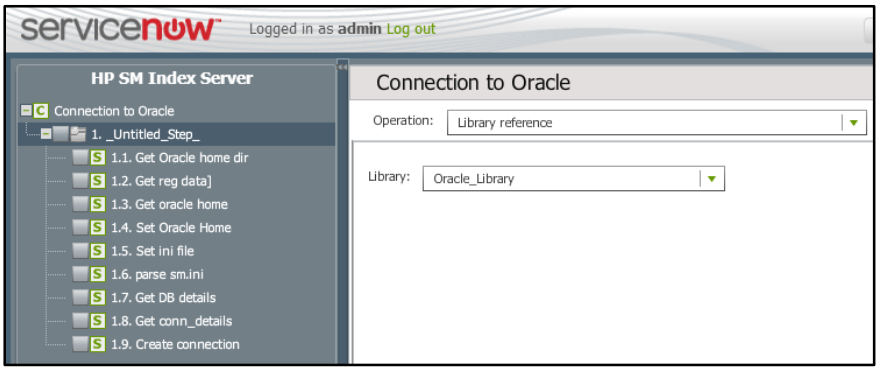
1. Display the steps you want to combine, select the checkbox for each of those steps, right-click any one of them, and select the **Create Library From Marked Steps** option.



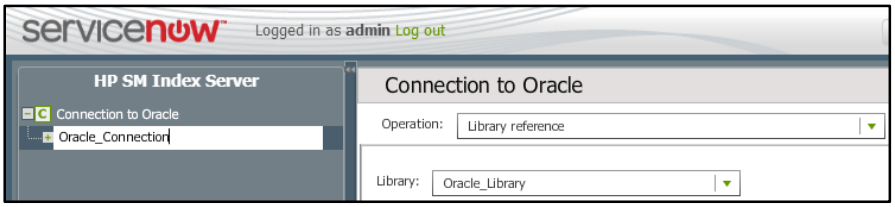
2. The **Create New Library** dialog box is displayed. Type in a unique **Library name** and a meaningful **Library description**, then click **OK**.



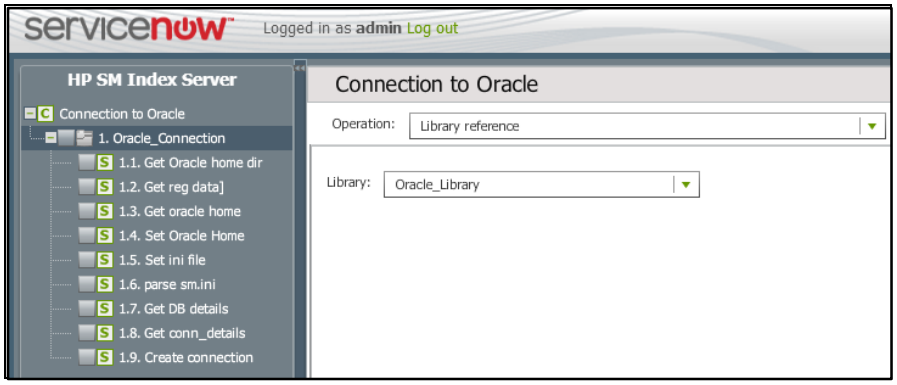
3. The Library reference operation displays the steps to be combined as substeps under an **_Untitled_Step_** node.



4. Double-click the **_Untitled_Step_** node and overwrite it with name you want to give to its collection of substeps.



5. You can click that node's [+] icon to display its substeps.



Match

Description

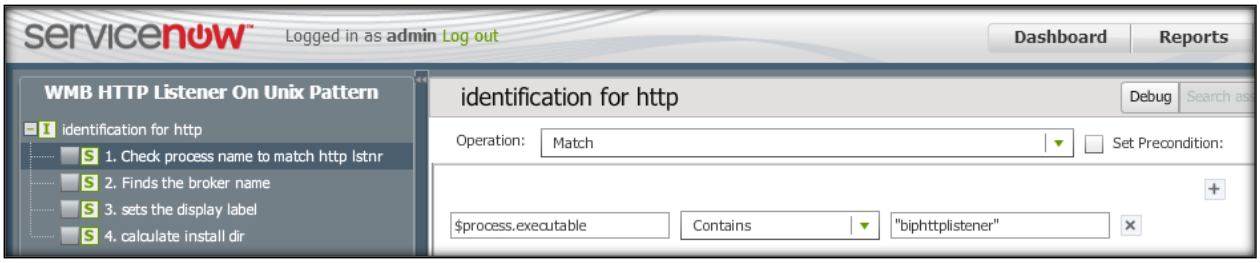
Purpose Enables the discovery process to continue if the specified conditions are met. Discovery terminates if the specified conditions are not met.

Instructions

1. Define the condition you want to check. For instructions on defining criteria, see DEFINING CONDITIONAL CRITERIA.
2. For each additional condition, click the + icon and fill in the condition and its AND/OR relationship. For data about specifying conditions, see DEFINING CONDITIONAL CRITERIA.

Example

Figure 17: Specifying criteria that determine if the Discovery process continues or stops



Example taken from:	
Hierarchy	Applications > Business Integration Software
CI Type	IBM WMB Http Listener
Pattern	WMB HTTP Listener On Unix Pattern
Section	Identification for http
Step # and Name	1. Check process name to match http lstnr

Merge Table

Description

Purpose Merge content from two source tables into a target table if specified criteria are satisfied.

Merge criteria can be specified as either field equalities or conditions:

- Field equality criteria are satisfied if both source tables have matching values in specified fields.
- Condition criteria are satisfied when both source tables satisfy specified condition criteria.

Note: Some of the fields in the screen change according to the type of merge criteria you selected.

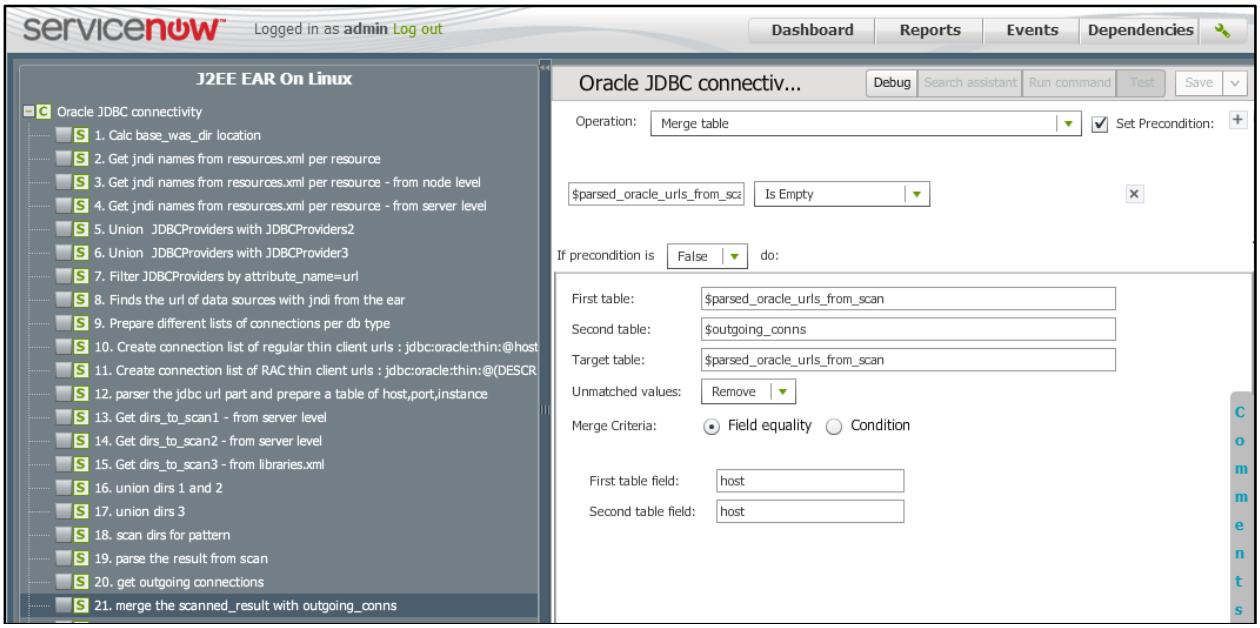
You can specify whether only the rows that satisfy the criteria should be merged, or if all rows in both tables should be merged if the criteria are satisfied.

Instructions

1. Specify the names of the two source tables in the **First table** and **Second table** fields.
2. Specify the name of the **Target** table.
3. Select a value from the **Unmatched values** drop-down list:
   - ✓ To merge all rows from both source tables into the target table if merge criteria are satisfied for any row, select **Keep**.
   - ✓ To merge only matching rows from both source tables into the target table and exclude non-matching rows, select **Remove**.
4. Select one of the **Merge Criteria** radio buttons:
   - ✓ If the merge should be based on fields that have matching values:
     i. Select **Field Equality**.
     ii. Specify which field in the first and second table should be compared to each other.
   - ✓ If the merge should be based on certain conditions being satisfied:
     i. Select **Condition**.
     ii. Specify the condition. For each additional condition, click the + icon, fill in the condition, and specify its AND/OR relationship. For information about specifying conditions, see DEFINING CONDITIONAL CRITERIA.

Example

Figure 18: Specifying criteria that determine if 2 source tables are merged into a target table



Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	Websphere EAR
Pattern	J2EE EAR On Linux
Section	DB2 JDBC connectivity
Step # and Name	21. merge the scanned_result with outgoing_conns

Parse Operations

Note: This section does *not* describe the **Parse URL** operation, which is described on page 60.

This section describes the following parsing operations:

- Parse Command Output
- Parse File
- Parse Variable

These operations are nearly identical. They differ basically in the source of the text to be parsed.

Description

Purpose Extract information from a source and place it into a variable table. The appropriate operation depends on the source.

When defining this operation, your first task is to specify the source of the data that will be parsed. Next, you need to identify the information to be extracted and the variable(s) into which the extracted information will be placed. There are a number of strategies available for parsing. The strategies do *not* depend on the chosen operation; that is, the same strategies are available regardless of the operation you select. One strategy is automatically selected by default, but you can override the default.

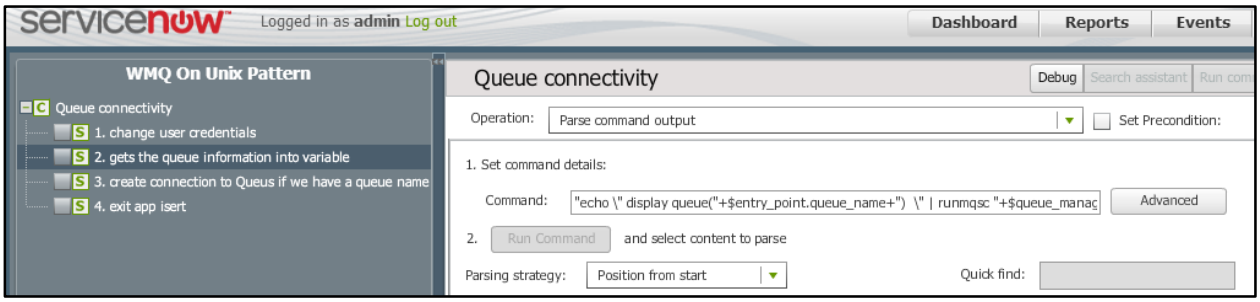
Some of the fields that are displayed and that must be filled in depend on the selected parsing strategy. Therefore, how you fill in these fields is described separately by parsing strategy.

Finally, at the end of the procedure, you can indicate whether to terminate the discovery process if the variable that will hold the extracted values is empty or cannot be found.

Instructions

1. Select the appropriate parsing operation.
   - ✓ If the source text comes from a file, select **Parse File**.
   - ✓ If the source text comes from a variable, select **Parse Variable**.
   - ✓ If the source text comes from the output of a command, select **Parse Command Output**.
2. Display the text source in the textbox, according to the selected operation. If the operation is
   - ✓ **Parse Command Output:**

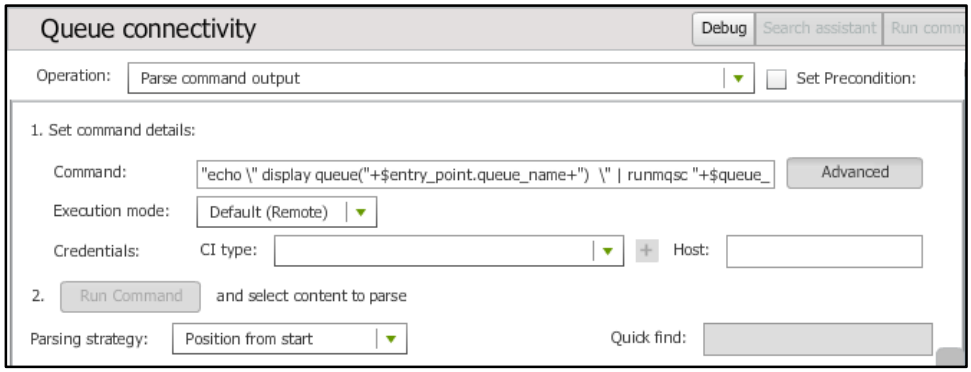
Figure 19: Parse Command Output procedure



i. Specify the command by starting to type it. As you type, AutoCompletion displays a command list. Select the appropriate command. You can concatenate multiple commands.

ii. To use different credentials and/or to use a different execution mode, click the Advanced button and do the following:

Figure 20: Parse command output – Advanced procedure



a. To change the **Execution mode**, select the desired mode from the drop-down list. The mode determines how communication with the remote machine is invoked. Valid values: **Default (Remote)**, **Windows Service**, **Local Script**.

b. To specify different Credentials:

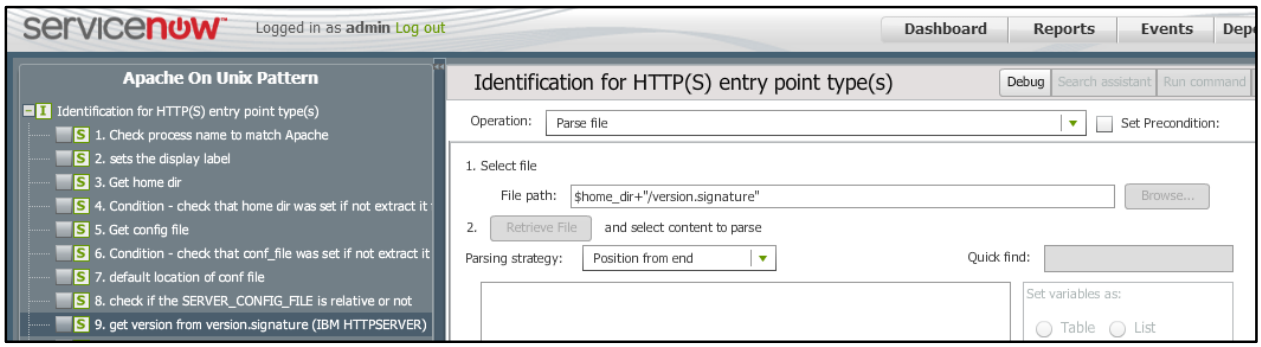
1. Select the **CI type** from its drop-down list.
2. To insert placeholders in the command for user name and password, place the cursor at the appropriate location in the **Command** field and click the + icon. Placeholders (format ***$$username$$*** and ***$$password$$***) are inserted.

   Note: Alternatively, you can insert these placeholders manually.
3. Specify the **Host**.

iii. Click the **Run Command** button to display the command output in the textbox.

✓ **Parse File:**

Figure 21: Parse File operation

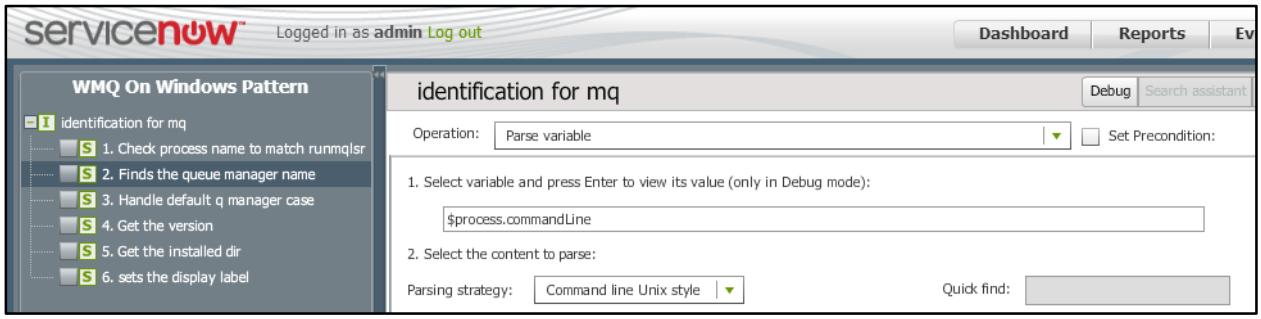


i. Either specify the full path and name of the file from which text will be parsed, or click the **Browse** button and select the file.

ii. Click the Retrieve File button to display the file text in the textbox.

✓ **Parse Variable:**

Figure 22: Parse variable operation



i. Specify the variable by starting to type it. As you type, AutoCompletion displays a variables list. Select the appropriate variable.

ii. Press Enter to display the variable's value in the textbox. (You must be in Debug mode).

The text is displayed in the textbox. The selection in the **Parsing strategy** field automatically changes so that it is appropriate to the text format. For example, if the textbox displays the contents of an XML file, the **Parsing strategy** field will automatically display the value XML File.

3. If desired, change the Parsing strategy. Some displayed fields change according to the parsing strategy.

4. Using the parsing strategy links and descriptions in TABLE 2 , click the appropriate parsing strategy in the table to access the instructions for filling in the parsing strategy details, then fill in the details. (Identify what will be extracted and into which variable(s) the extracts will be placed.)

   The displayed fields, and instructions for filling them in, vary according to the selected parsing strategy. However, the same general choice is offered for identifying the text to be extracted, regardless of parsing strategy:

   ✓ You can select the text in the textbox. This method only works in Debug mode. In this case, all corresponding matching text in the textbox is highlighted and color coded, allowing you to distinguish between different text selections and their variables.

   Note: To locate specific text in a textbox, instead of scrolling through the text, you can type the text in the **Quick Find** field. A search is performed and refined as you type.

   ✓ You can fill in parameters that will identify the text. (For example, you might fill in a delimiter and start position.) This method works whether you are in Debug mode or not.

   TABLE 2 lists each parsing strategy and describes its purpose.

   Note: Each strategy in the list is also a link to instructions for filling in the details of the strategy. Several strategies can have the same set of instructions.

Table 2: Parsing Strategies

Link to Detailed Description	Brief Description
DEFINING DETAILS for Keyword, Command, and Positional Type Parsing Strategies	Retrieve text directly following a specific keyword
DEFINING DETAILS FOR KEYWORD, Command, and Positional Type Parsing Strategies	Retrieve the value of a command line parameter using Java style parameters
DEFINING DETAILS FOR KEYWORD, Command, and Positional Type Parsing Strategies	Retrieve the value of a command line parameter using standard Unix/Windows parameters
DEFINING DETAILS for the Delimited Text Parsing Strategy	Retrieve text specified by delimiters and position within the line (This is the most common way to retrieve text from generic text files.)
INI FILE	Retrieve text from a .INI file
DEFINING DETAILS for Ini, Oracle, Properties and XML file Parsing Strategies	Retrieve text from a .ORA file (used by various Oracle products)
DEFINING DETAILS FOR Keyword, Command, and Positional Type Parsing Strategies	Retrieve text specified by its position from the end of the line
DEFINING DETAILS FOR Keyword, Command, and Positional Type Parsing Strategies	Retrieve text specified by its position from the beginning of the line
DEFINING DETAILS for Ini, Oracle, Properties and XML file Parsing Strategies	Retrieve text from a .properties file (common in the Java world)
DEFINING DETAILS for the Regular Expression Parsing Strategy	Retrieve text specified by a regular expression
DEFINING DETAILS for the Vertical File Parsing Strategy	Retrieve text from a structured text file where each set of data spans multiple lines
DEFINING DETAILS for Ini, Oracle, Properties and XML file Parsing Strategies	Retrieve text from a .XML file

Defining Details for Ini, Oracle, Properties and XML file Parsing Strategies

This section describes how to parse text into variables for all File-type parsing strategies except **Vertical File**. Follow these instructions if you selected one of these parsing strategies: **Ini file**, **Oracle file**, **Properties file**, or **XML file**.

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

The simplest way to do this is to display the content of a file in the textbox and select the relevant string. For each string you select, its position and delimiters relative to its context are stored. This enables the same definitions to apply to other files with the same structure even though the text varies.

However, this method works only in Debug mode. And it selects the entire text within a context. For example:

In an XML file with the following line:
<ciTypeID>123-456-7890000000</ciTypeID>,
if you try to select only **456** in the textbox, the entire string between the keywords will be selected.

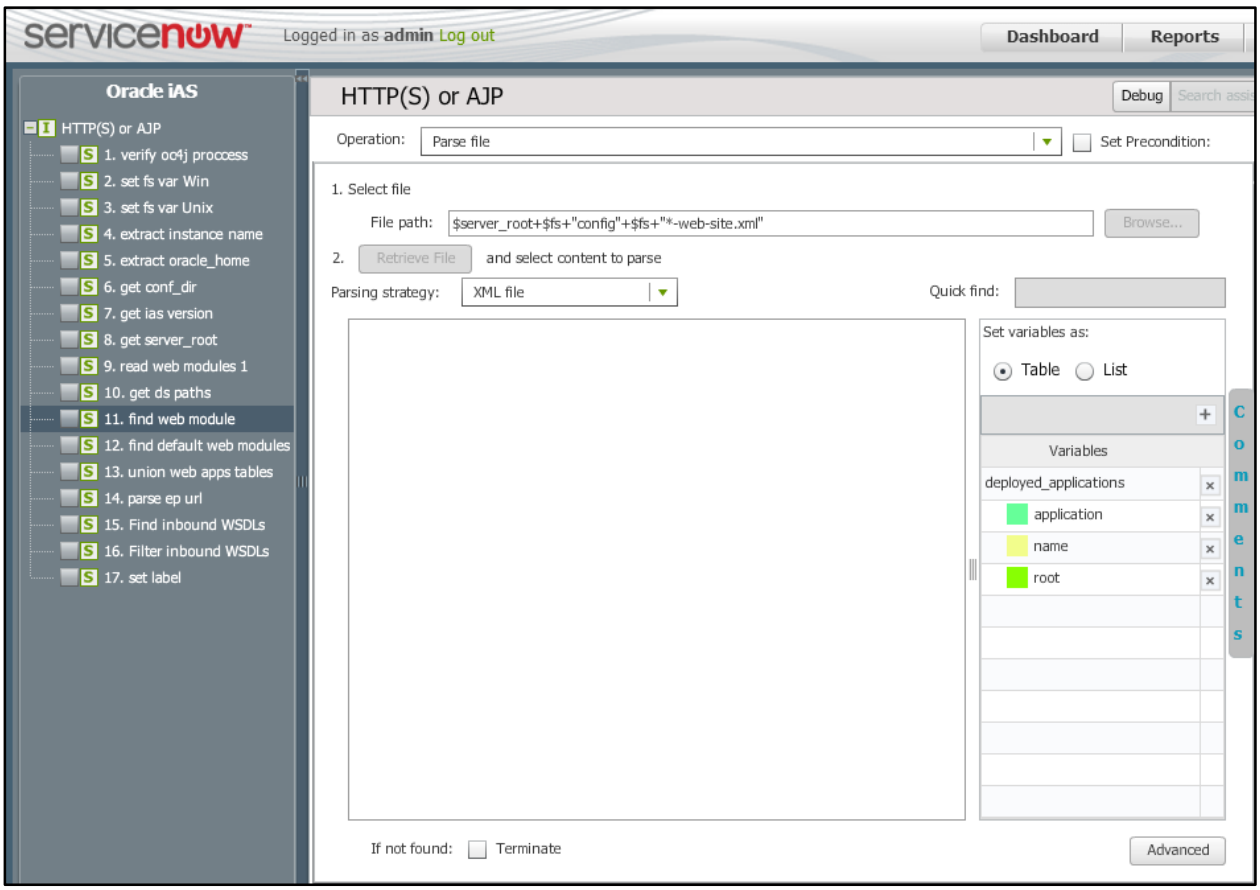
Therefore, an **Advanced Parsing Options** dialog box enables you to specify a delimiter and position for identifying the text string. Use this dialog box if you are not in debug mode, or to make a more refined selection than you can make in the textbox. For example, you could refine the above selection in the **Advance Parsing Options** dialog box by specifying a delimiter (**-** in this case) and the number of positions to extract after the delimiter (**3** in this case) in order to extract the string (**456** in this case).

1. If you are in Debug mode, do the following for each string to be parsed:
   a. Select the string in the textbox. (The cursor becomes a "brush" in the textbox.)

      All matching strings in the same context are automatically selected and color coded. The **Define Variable Name** dialog box enables you to assign the selected string to a variable.

   b. Provide a unique and meaningful name to the variable and click **OK**. The variable with a matching color coded box is displayed in the **Variables** table to the right of the textbox.

Figure 23: Assigning color-coded strings to same color-coded Variables

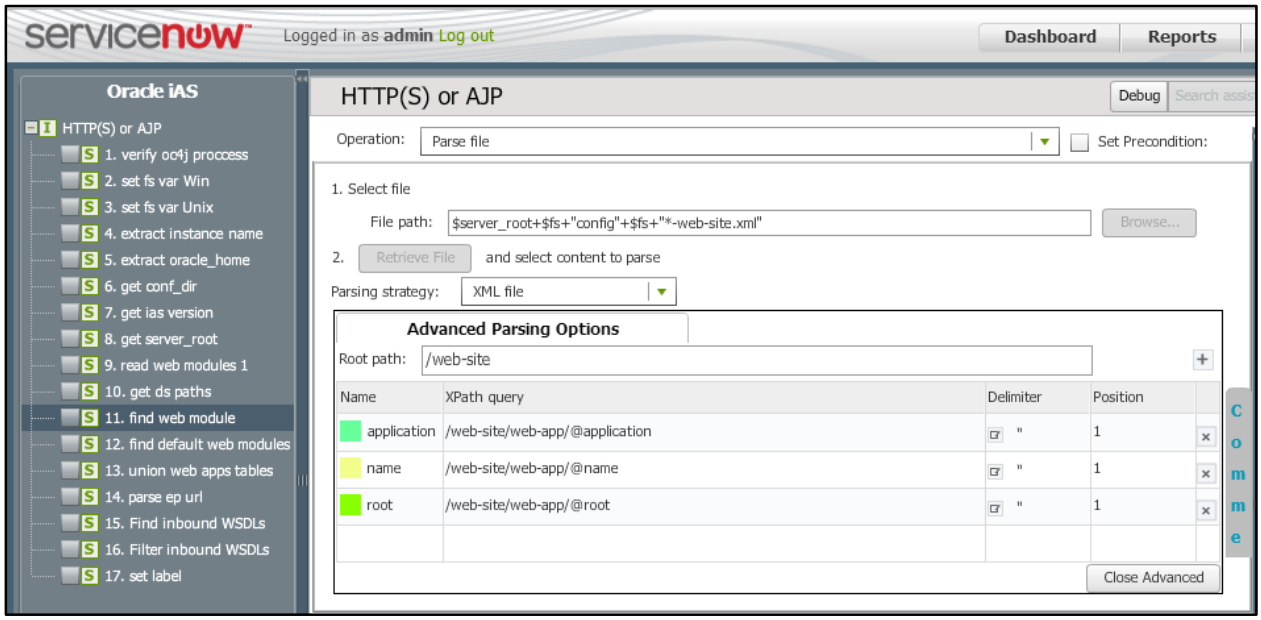


   c. To refine strings you previously selected in the textbox, do the following:

Note: You can identify additional strings and variables by clicking the + icon to add a string and its variable.

i. Click Advanced. The **Advanced Parsing Options** dialog box opens. It displays the defined variables in the **Name** field and the string in the **XPath query** field.

Figure 24: Advanced Parsing Options



For each string that you want to define:

ii. Define the delimiter.

iii. Define the position.

d. When done, click Close Advanced.

2. If you are not in Debug mode, to identify strings and the variables that should contain them, do the following:

a. Click Advanced. The same **Advanced Parsing Options** dialog box opens. (See above.)

b. In the **Root path** field, specify the section (the hierarchical branch in the file structure) where the parsing takes place.

c. For each string/variable that you want to identify:

i. Click the + icon.

ii. Specify the column name in the **Name** field.

iii. Specify the **XPath query**.

iv. Define the **Delimiter**.

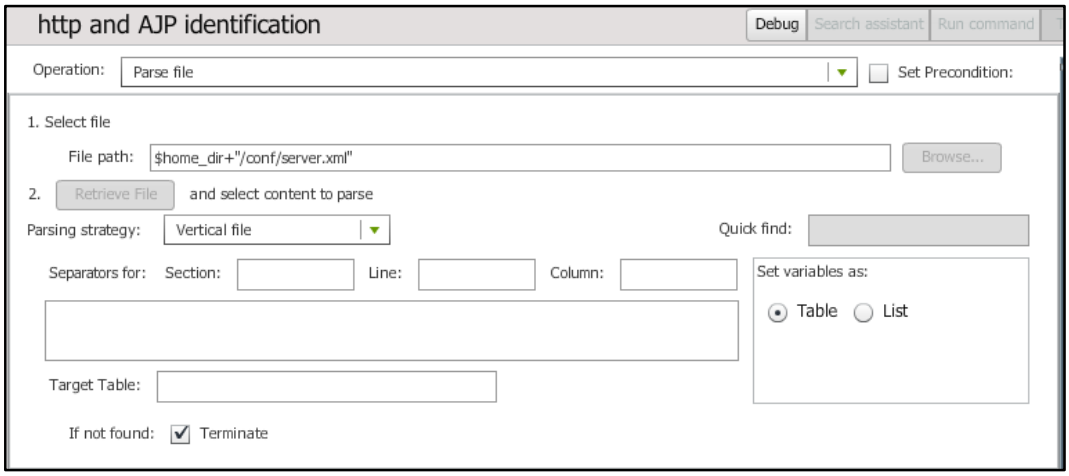
v. Define the **Position**.

d. When done, click Close Advanced

Defining Details for the Vertical File Parsing Strategy

This section describes how to parse text into variables using the **Vertical file** parsing strategy.

Figure 25: Vertical File Parsing Strategy



1. Identify the separators used between sections, lines and columns. The text area displays the text with the results highlighted.
2. Select the **Table** or **List** (scalar) radio button. For the difference between table and scalar variables, see Working with Variables on page 21. (As a general rule, if the Target Table contains multiple columns, select the **Table** radio button.)
3. Specify the name of the **Target Table**.
4. If the discovery should stop if no values are extracted, select (check) the **Terminate** checkbox.

Defining Details for Keyword, Command, and Positional Type Parsing Strategies

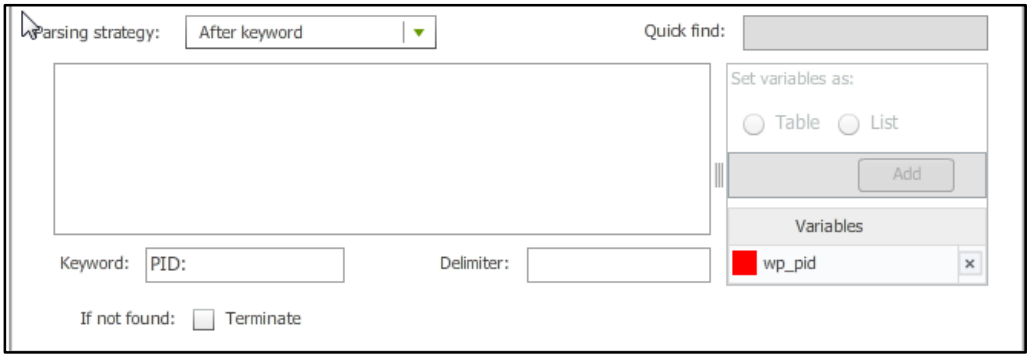
This section describes how to parse text into a variable for the following parsing strategies:

- After Keyword
- Command Line Java Style
- Command Line Unix style
- Position from Start
- Position from End

These strategies are generally used to extract a value from a variable. They allow you to define only a single value for extraction. To extract multiple strings, define multiple steps.

You can identify the text to be parsed either by selecting it in the textbox, or by specifying relevant identifying information such as a delimiter and keyword or start position or end position, depending on the operation. The matching text will automatically be highlighted.

Figure 26: Keyword, Command & Positional Parsing Strategies



1. If you are in Debug mode, identify the string to be extracted and the variable that should contain it, as follows:
   a. Select the string to be extracted.

      The string's identifying Keyword (or Position from Start or Position from End) value and its delimiter are automatically filled in and the **Define Variable Name** dialog box opens allowing you to assign the selected string to a variable.
   b. Provide a unique and meaningful name for the variable and click **OK**. The variable is displayed in the **Variables** table to the right of the textbox.
2. If you are not in Debug mode, identify the string to be extracted and the variable that should contain it, as follows:
   a. Specify the Keyword (or Position from Start or Position from End) value and the delimiter in the appropriate fields.
   b. Add the variable by clicking the + icon by the **Variables** table and assign a name to the variable.
3. To terminate the discovery process if the variable that will hold the extracted values is empty or cannot be found, select (check) the **Terminate** checkbox.

Defining Details for the Regular Expression Parsing Strategy

This section describes how to parse text into variables using the Regular Expression query language as the parsing strategy. To use this strategy, you must know how to use this query language.

You can only specify a single expression, but depending on its wording, multiple values (identified by being enclosed in parentheses) can be extracted. Matching of variables to parentheses goes according to the order of the parentheses sets. The first variable is matched to the first set of parentheses, etc.

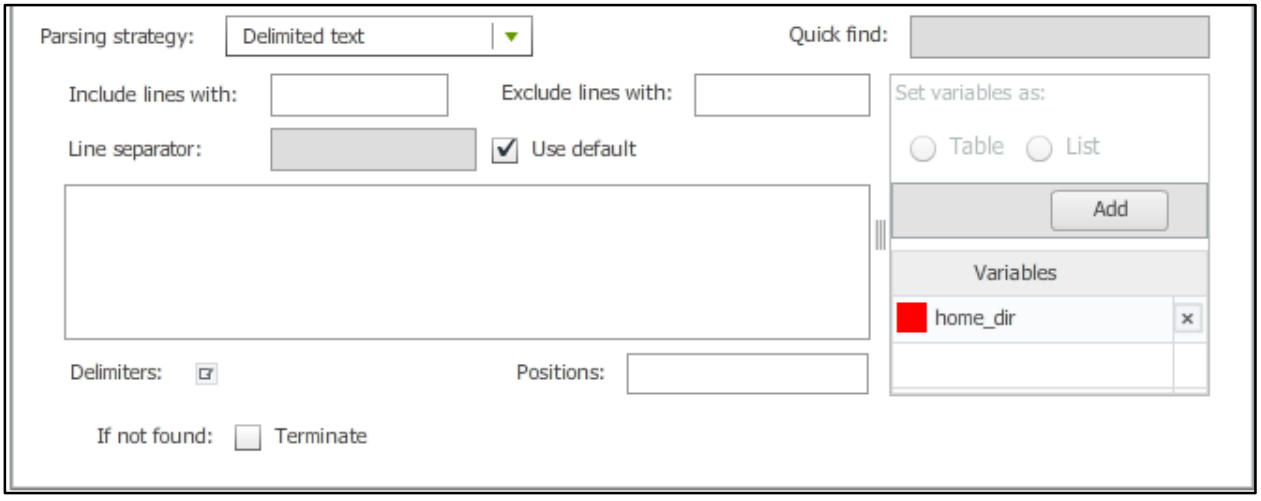
Figure 27: Regular Expression Parsing Strategy

1. Specify the expression in the **Regular expression** text box. The matching text is highlighted in the textbox. You cannot select text in the textbox using this parsing strategy.
2. Select the **Table** or **List** (scalar) radio button. For the difference between table and scalar variables, see Working with Variables on page 21.
3. For each set of parentheses in the expression, in sequence, click the + icon and assign a variable name.
4. To terminate the discovery process when a variable that will hold an extracted value is empty or cannot be found, select (check) the **Terminate** checkbox.

Defining Details for the Delimited Text Parsing Strategy

This section describes how to parse text into variables using the **Delimited text** parsing strategy.

Figure 28: Delimited Text Parsing Strategy



1. In the **Include lines with** and **Exclude lines with** fields, specify any constants that should be used to determine if lines are included or excluded from the text selection.
2. If the text uses a non-default character as a line separator (default separators are either NEWLINE or CARRIAGE RETURN, depending on the operating system), uncheck the **Use default** checkbox and specify the character that is used in the **Line separator** field.

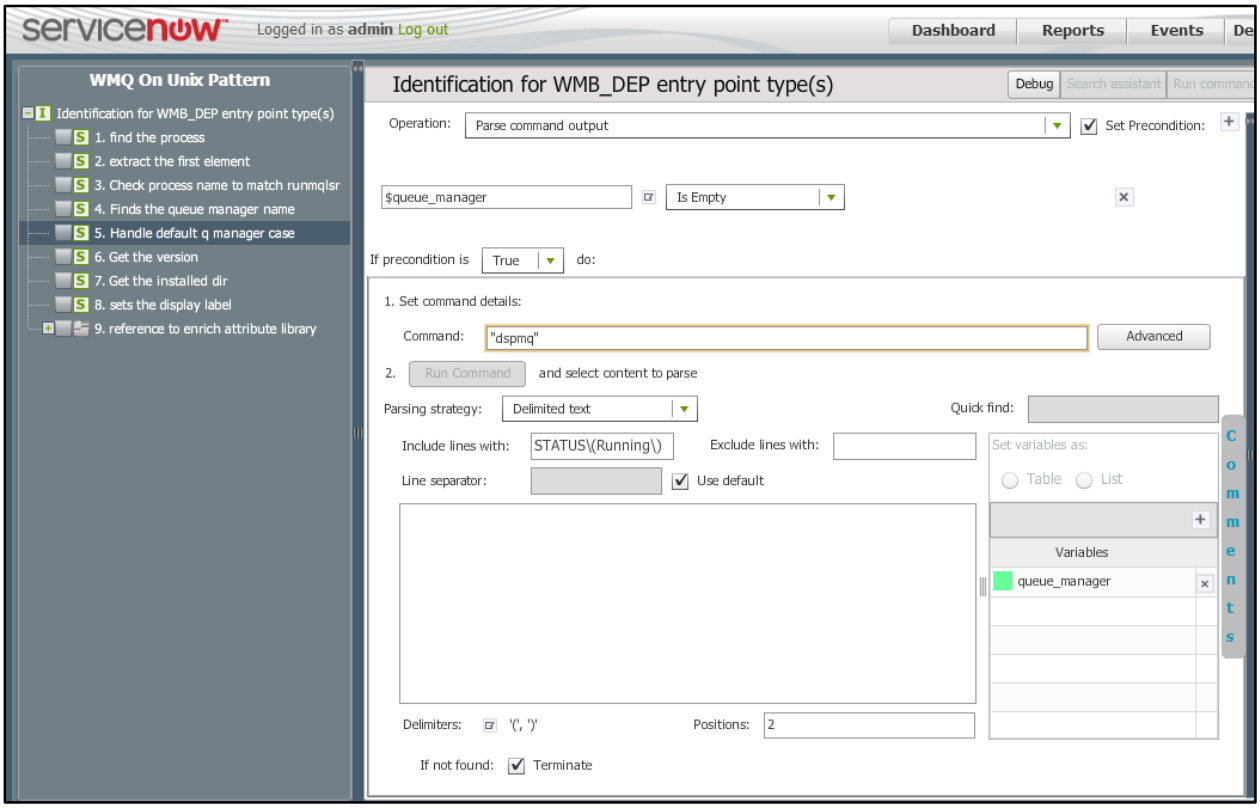
3. If you are in Debug mode, select the string and define its variable as follows:
   a. Select the string in either of the following ways:
      - Type the string in the **Quick find** field and press Enter.
      - Select the string in the textbox. (The cursor becomes a "brush" in the textbox.)

      All identical strings in the textbox are automatically selected and color coded and the **Define Variable Name** dialog box opens.
   b. In the **Define Variable Name** dialog box, assign a unique and meaningful name to the variable and click **OK**.

      The variable, accompanied by a matching color-coded box, opens in the **Variables** table to the right of the textbox.
4. If you are not in Debug mode, identify the string and define the variable as follows:
   a. To define the delimiters:
      i. Click the **Delimiters** icon. The **Edit delimiters** dialog box opens.
      ii. Fill in the delimiter details.
      iii. For each additional delimiter, click the  icon and fill in its details.
      iv. When done, click **OK**.
   b. In the **Positions** textbox, specify the position(s). Separate multiple positions by commas.
   c. Add the variable by clicking the  icon by the **Variables** table and assign a name to the variable.
5. If discovery should stop if no values are extracted, select (check) the **Terminate** checkbox.

Example: Parse Command Output

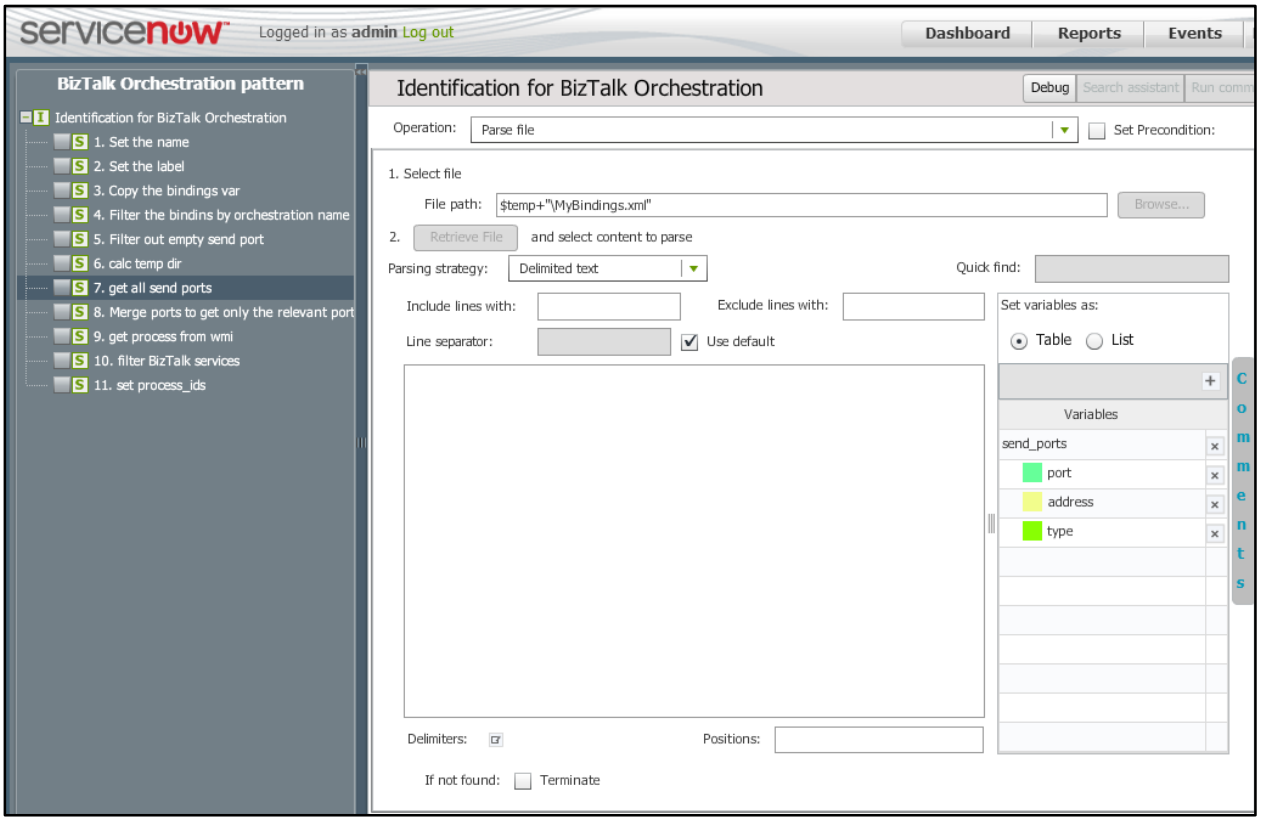
Figure 29: Parse Command output



Example taken from:	
Hierarchy	Business Integration Software
CI Type	IBM WebSphere MQ WebSphere EAR
Pattern	WMQ On Unix Pattern
Section	Identification for WMB_DEB entry point type(s)
Step # and Name	5. Handle default q manager case

Example: Parse File

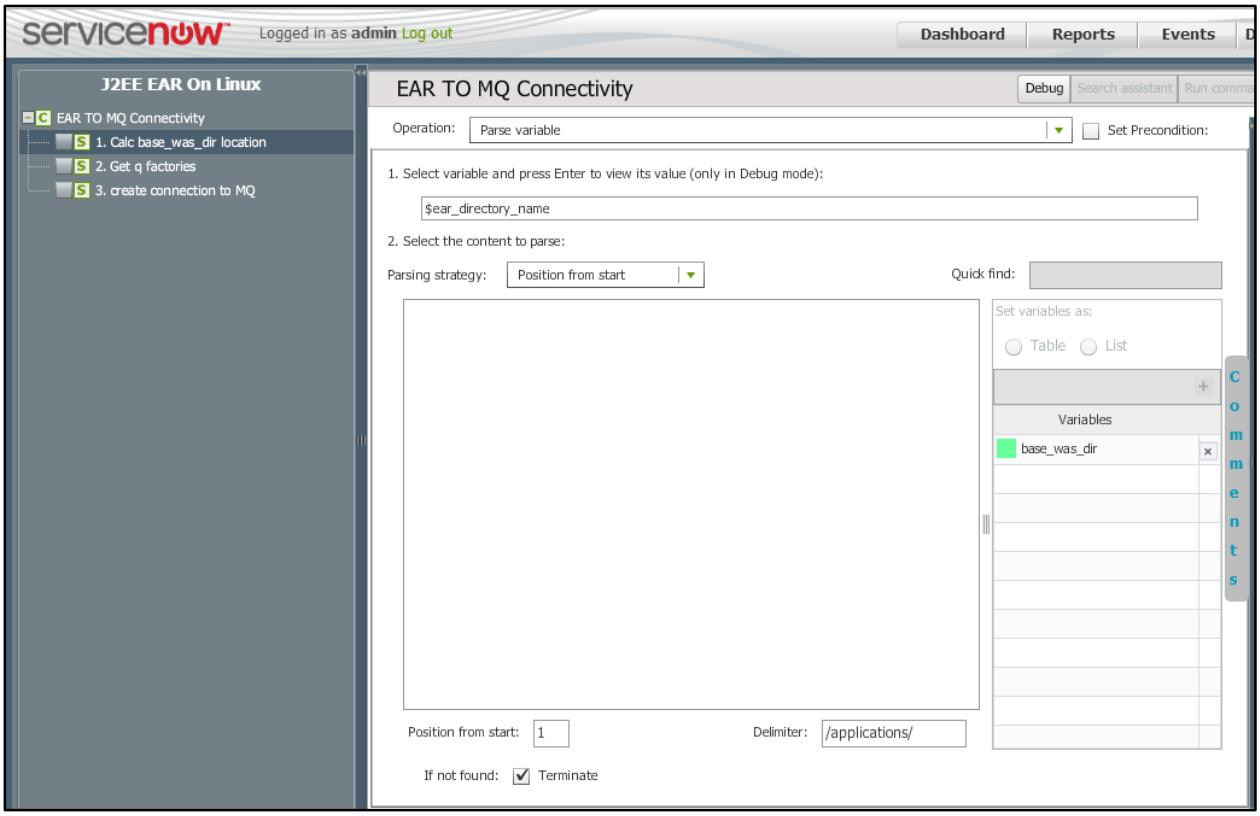
Figure 30: Parse File



Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	WebSphere EAR
Pattern	BizTalk Orchestration pattern
Section	Identification for BozTalk Orchestration
Step # and Name	7. get all send ports

Example: Parse Variable

Figure 31: Parse Variable



Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	WebSphere EAR
Pattern	J2EE EAR on Linux
Section	EAR TO MQ Connectivity
Step # and Name	6. Calc base_was_dir location

Parse URL

Description

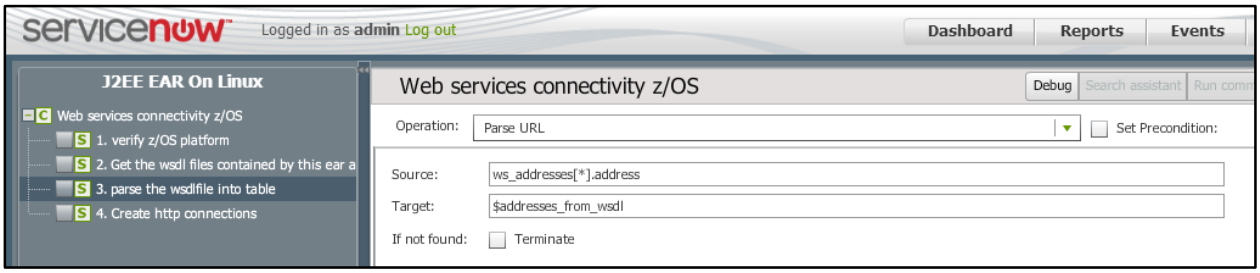
Purpose Break down a URL into its components

Instructions

1. In the **Source** field, specify the pattern of the URLs.
2. In the **Target** field, specify the table into which the results will be placed.
3. If discovery should terminate if no results are found for the query, select (check) the **Terminate** checkbox.

Example

Figure 32: Parse URL



Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	Websphere EAR
Pattern	J2EE EAR on Linux
Section	Web Services Connectivity
Step # and Name	3. parse the wsdlfile into table

Put File

Description

Purpose Transfer a file from the collector to a remote computer.

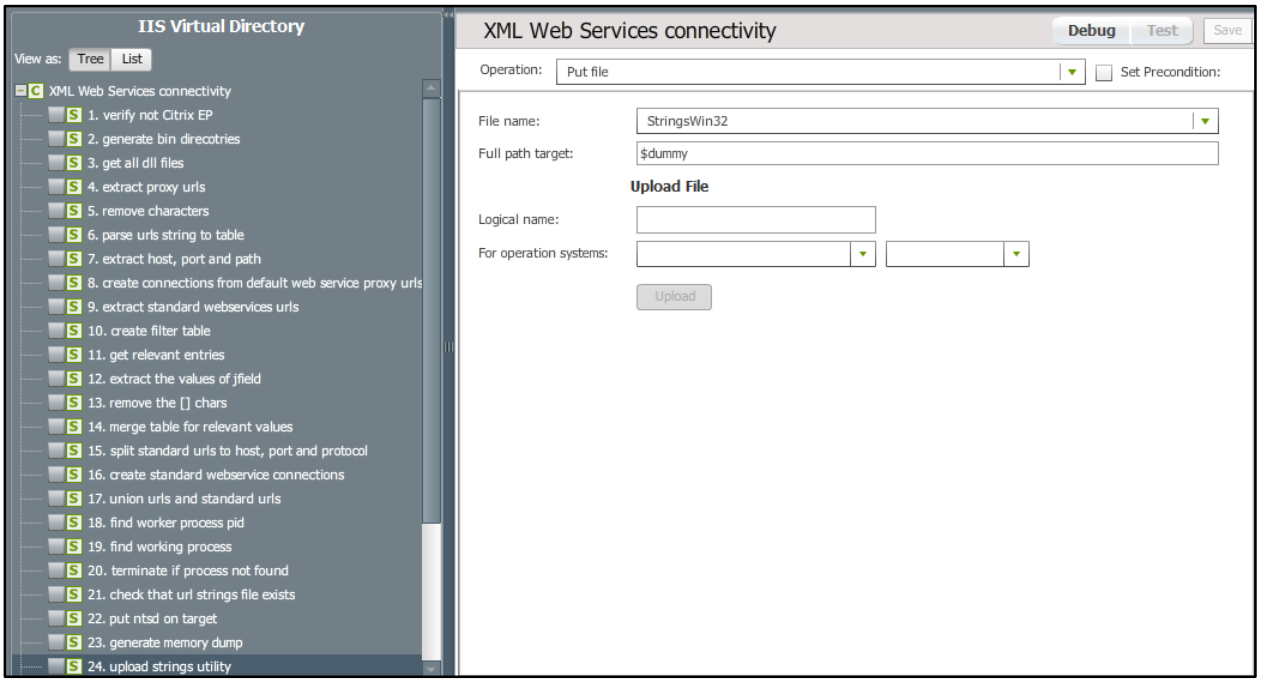
Instructions

Note: Before using this operation, ensure that you defined a logical name for your files.

1. In the **File name** field select the logical name of the file from the drop-down list or copy it from the collector.
2. Specify the **Full path target** on the remote computer.

Example

Figure 33: File transfer from a collector to a remote computer



Example taken from:	
Hierarchy	Applications > Web Servers
CI Type	Website
Pattern	IIS Website
Section	XML Web Services Connectivity
Step # and Name	9. Upload strings utility

Set Variable

Description

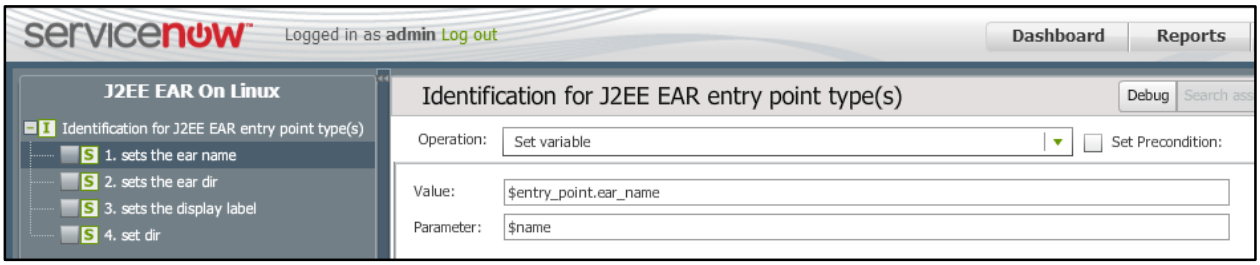
Purpose Apply a value to a variable.

Instructions

1. Specify the **Value**. Values can be variables or constants.
2. Specify the variable or table name in the **Parameter** field.

Example

Figure 34: Assigning a value to variables or constants



Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	Websphere EAR
Pattern	J2EE EAR on Linux
Section	Identification for J2EE EAR entry point
Step # and Name	1. sets the ear name

SNMP Query

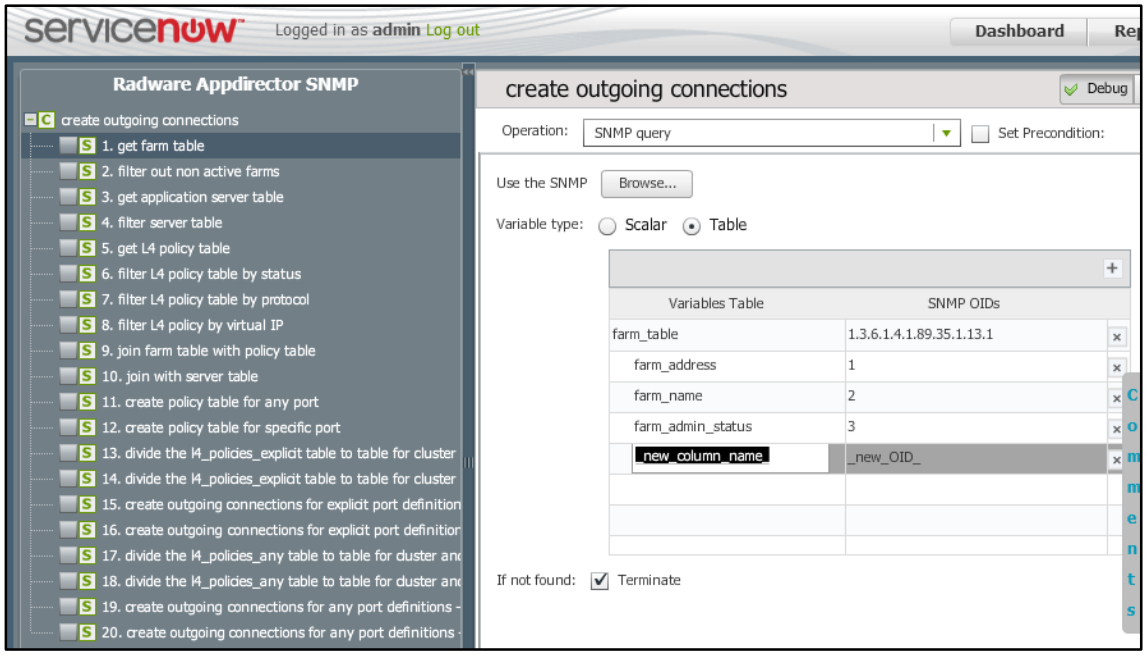
Description

Purpose Execute an SNMP query.

Instructions

Do either of the following:

- To manually define the variable(s) and SNMP OID(s) for the query:
  1. Select the **Variable type** (Scalar *or* Table). The display changes according to your selection.
  2. If you select the **Scalar** radio button, a single set of fields is displayed. Fill in the **Variable** name and **SNMP OID**.
  3. If you select the **Table** radio button, a table is displayed.

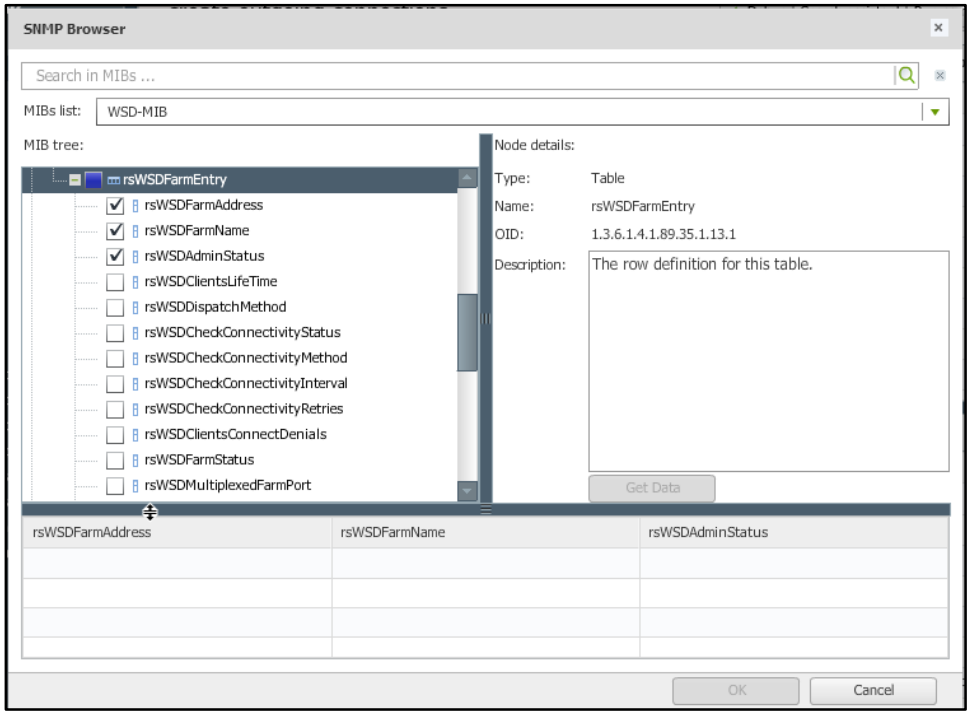


  4. For each column to add to the query:
     a. Click the + button.
     b. Replace **_new-column-name_** with the actual column name in the **Variables Table** column.
     c. Replace **_new_OID_** with the actual OID in the **SNMP OIDs** column.
  5. If discovery should terminate if no results are found, select the **Terminate** checkbox.
- To use the Browse feature to help you find the variables and SNMP OIDs for the query:

  Note: You must be in **Debug** Mode to open the Browse window and use the Get Data request.

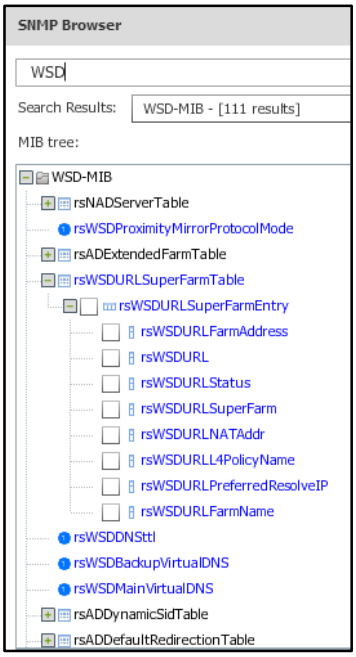
  1. Click the Browse... button. The **SNMP Browser** dialog box opens.

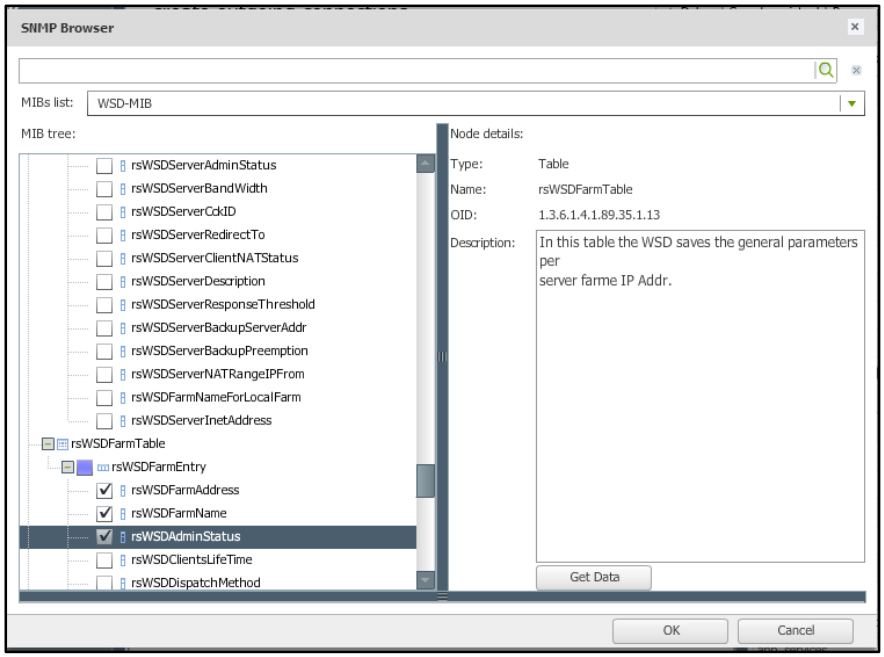
Figure 35: SNMP Browser dialog box



2. This dialog box enables you to browse in either of two ways. Do either of the following:
   - ✓ Select the MIB in the MIBs drop-down list. The contents of the selected MIB are displayed in the tree on the left side of the Browser dialog box. *or*
   - ✓ Perform a search for the MIB based on values as follows:

   a. In the **Search in MIBs** field, specify the search value and press the Enter key. A list containing the search results is displayed.

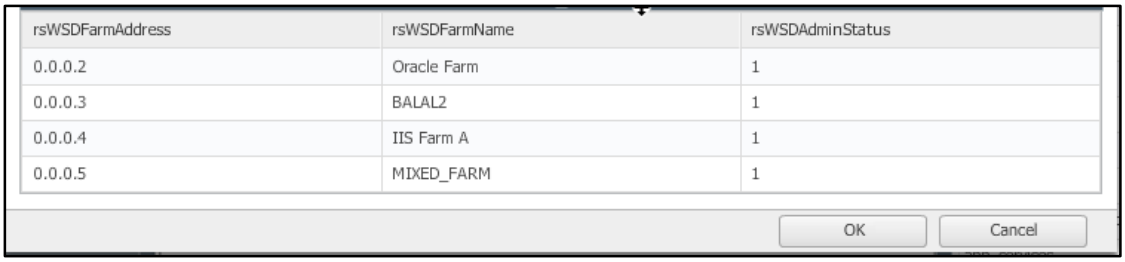




b. Select the desired result. The contents of the MIB that contains the selected results value is displayed in the tree on the left side of the **Browser** dialog box and the selected value is highlighted in the tree.

3. In the tree structure, select either a single scalar value, or select multiple columns from the same table. When selecting multiple columns in a MIB, the columns must be in the same table.
4. Click the Get Data button to retrieve the data for the selected variables. The data is displayed in the bottom panel of the **SNMP Browser** dialog box. In this example:

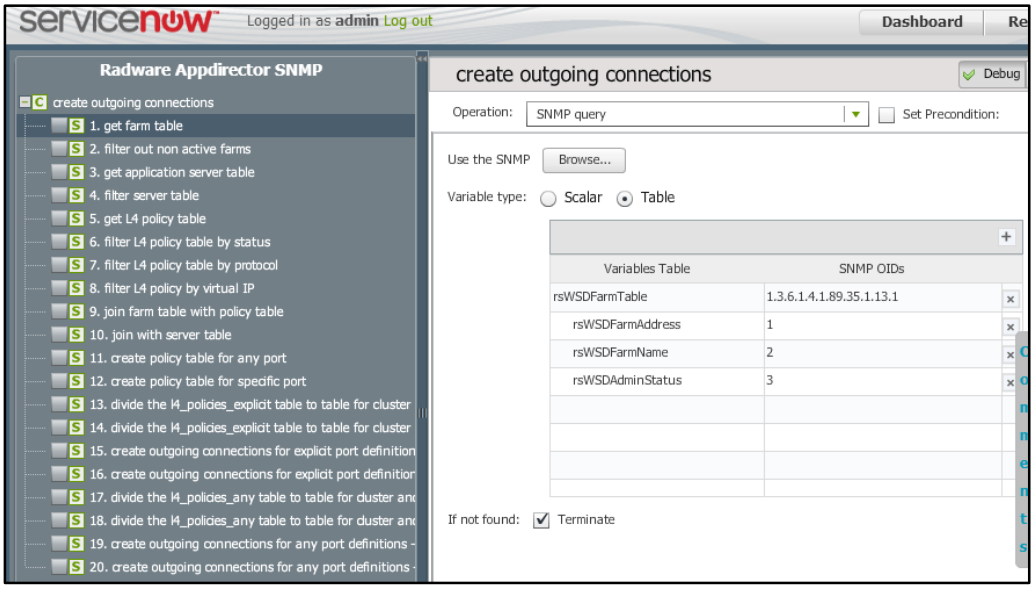
Figure 36: Bottom panel of the SNMP Browser dialog box



5. Click OK. The **SNMP Browser** dialog box closes and the SNMP Query operation window displays the relevant variables and SNMP OIDs.

Example

Figure 37: SNMP Query operation window



Example taken from:	
Hierarchy	Network > Load Balancers
CI Type	Radware Load Balancer
Pattern	Radware Appdirector SNMP
Section	create outgoing connections
Step # and Name	1. get farm table

Transform Table

Description

Purpose Add one or more computed columns to an existing table and place the results in a target table (which can be the same table).

Instructions

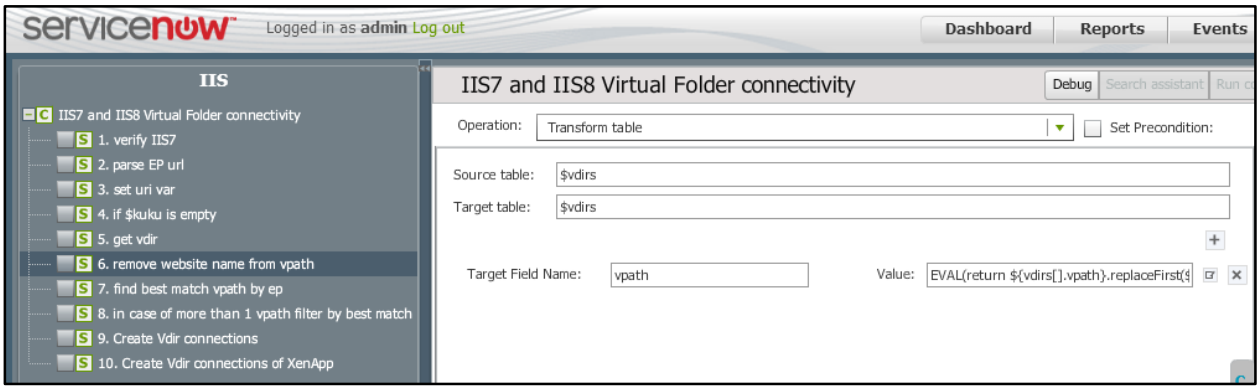
1. Specify the names of the source table and the target table.

   Note: These names are identical if you are adding columns to the source table.

2. For each column to be added to the target table:
   a. Click the + icon to add a line for specifying the column details.
   b. Specify the name to be assigned to the column in the **Target Field Name** field.
   c. In the **Value** field, specify the **operation** (expression) that determines the values added to the column.

Example

Figure 38: Specifying how to add computed columns to a table to produce a target table



Example taken from:	
Hierarchy	Applications > Web Servers
CI Type	IIS
Pattern	IIS
Section	IIS7 and IIS8 Virtual Folder connectivity
Step # and Name	6. remove website name from vpath

Unchange User

Description

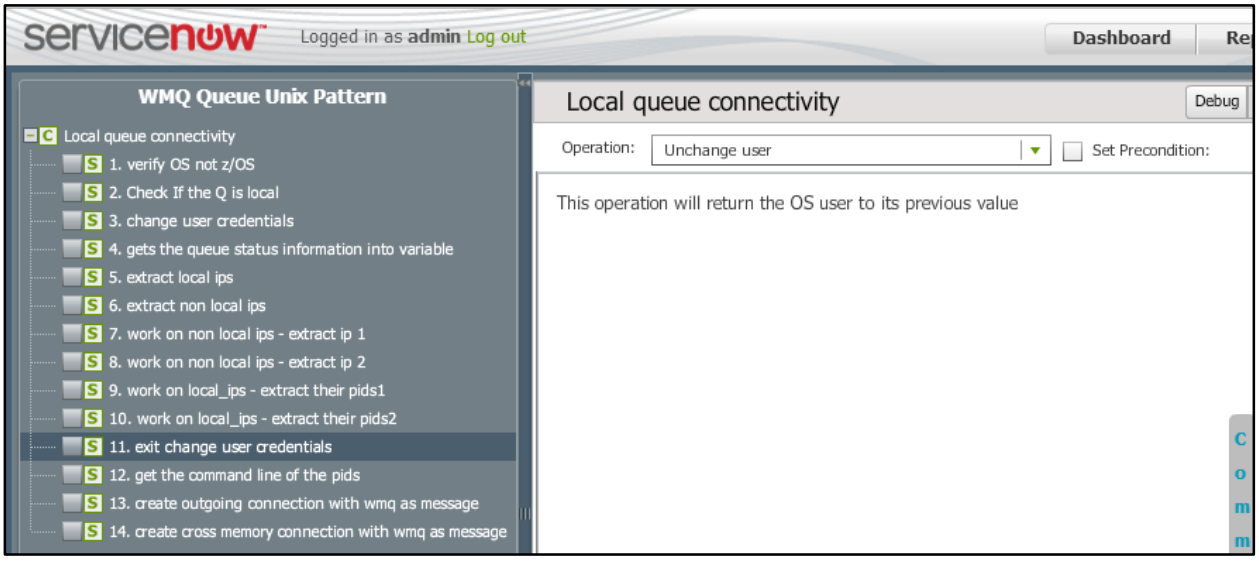
Purpose Switch back to the default Administrative credentials (if you previously performed a **Change User** operation).

Instructions

Just request the operation.

Example

Figure 39: Unchange user procedure



Example taken from:	
Hierarchy	Applications > Business Integration Software
CI Type	IBM WebSphere MQ Queue
Pattern	WMQ Queue Unix Pattern
Section	Local queue connectivity
Step # and Name	11. exit change user credentials

Union Table

Description

Purpose Append two tables that have the same format.

If two tables have the same format, you can append the second table to the end of the first table and place the results in a target table.

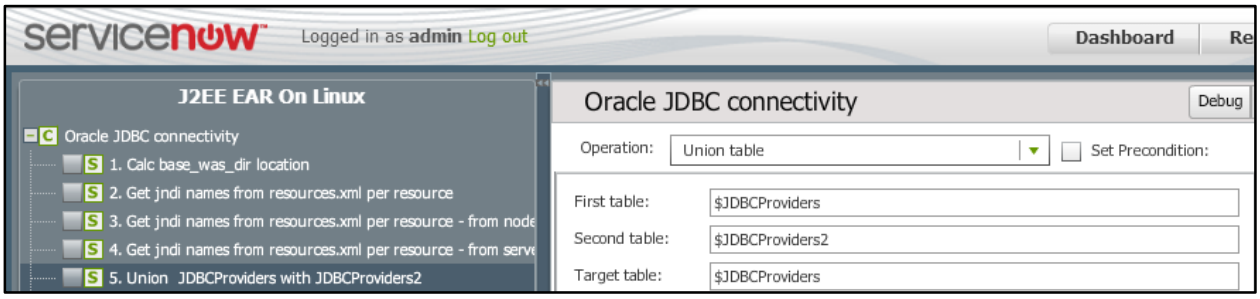
Note: You can optionally specify one of the source tables as the target table.

Instructions

1. Specify the names of the **First table**, the **Second table** and the **Target table**.

Example

Figure 40: Combining 2 tables with the same format to produce a target table



Example taken from:	
Hierarchy	Applications > Application Servers
CI Type	WebSphere EAR
Pattern	J2EE EAR on Linux
Section	Oracle JDBC connectivity
Step # and Name	5. Union JDBCProviders with JDBCProviders2

WMI Method Invocation

Description

Purpose Execute a method using WMI.

This operation allows you to select and execute a method that you select from a table that is returned by a WMI query (WMI QUERY) operation.

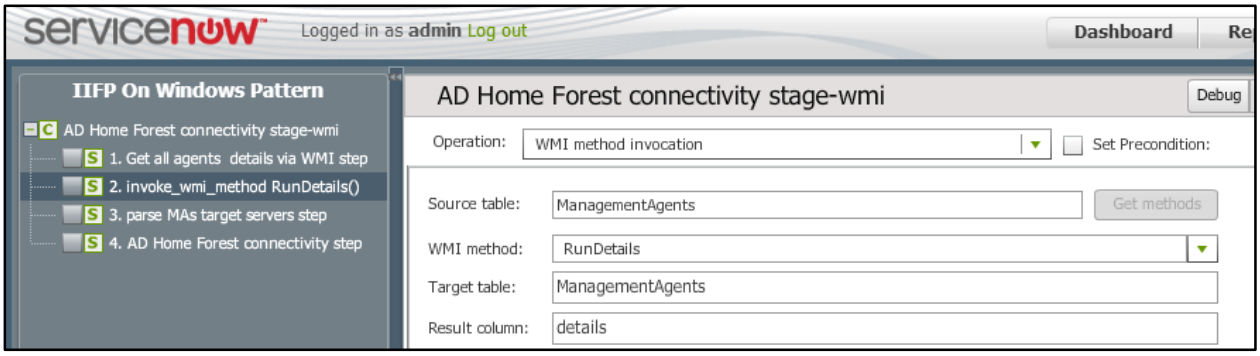
Notes: The source table must come from a WMI query operation.
Be sure to run a WMI query before running this step.

Instructions

1. Select the table that resulted from a WMI query as your source table.
2. Click the **Get methods** button.
3. Select the desired method from the **WMI method** drop-down list.
4. Specify the **Target table**.
5. In the **Result column** field, specify the name of the column that will contain the results of the method invocation.

Example

Figure 41: Processing a table returned by a WMI query



Example taken from:	
Hierarchy	Applications > Directory Services
CI Type	IIFP
Pattern	IIFP On Windows Pattern
Section	AD Home Forest connectivity stage-wmi
Step # and Name	2. Invoke WMI Method Run Details()

WMI Query

Description

Purpose Execute a WMI query on a remote Windows machine.

You can define the query statement in either of two ways:

- Explicitly define the query statement
- Fill in displayed fields and ServiceWatch will automatically generate the query statement based on the values you specify

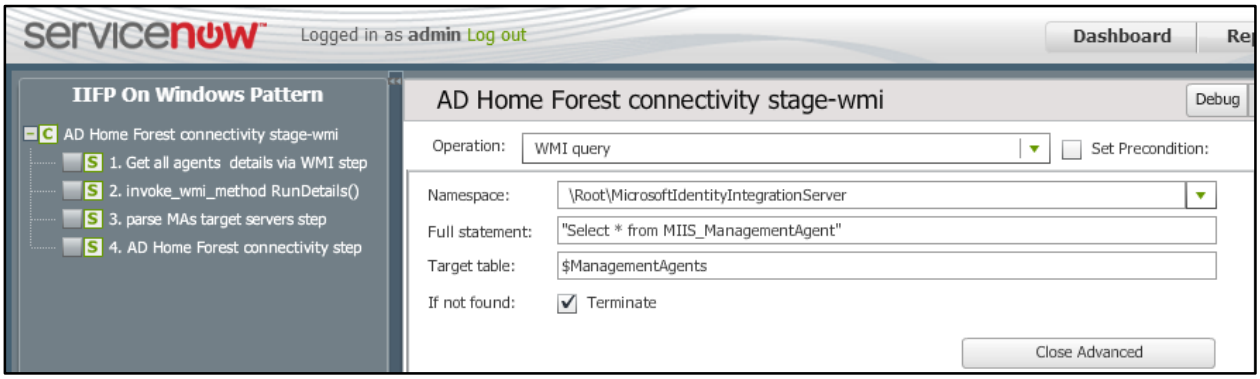
Regardless of method used, you must specify the **Target table** where the query results will be placed.

Instructions

1. Fill in the details in either of the following ways:

   ✓ To explicitly specify the query statement, click the Advanced button and specify values for all of the fields in the dialog box:

Figure 42: WMI Query Advanced dialog box



   i. Select the **Namespace** path from its drop-down list.

   ii. Specify the **Full statement** for the query.

   iii. Specify the **Target table** name.

   iv. If discovery should stop if no results are found, select (check) the **Terminate** checkbox.

   v. Click the Close Advanced button.

   ✓ To let ServiceWatch generate the query statement, do *not* click the Advanced button.

   i. Select the **Namespace** path from its drop-down list.

   ii. Search for or select the **Table name**. Its fields are displayed in the **Fields** area.

   iii. Select (click) the relevant fields in that area.

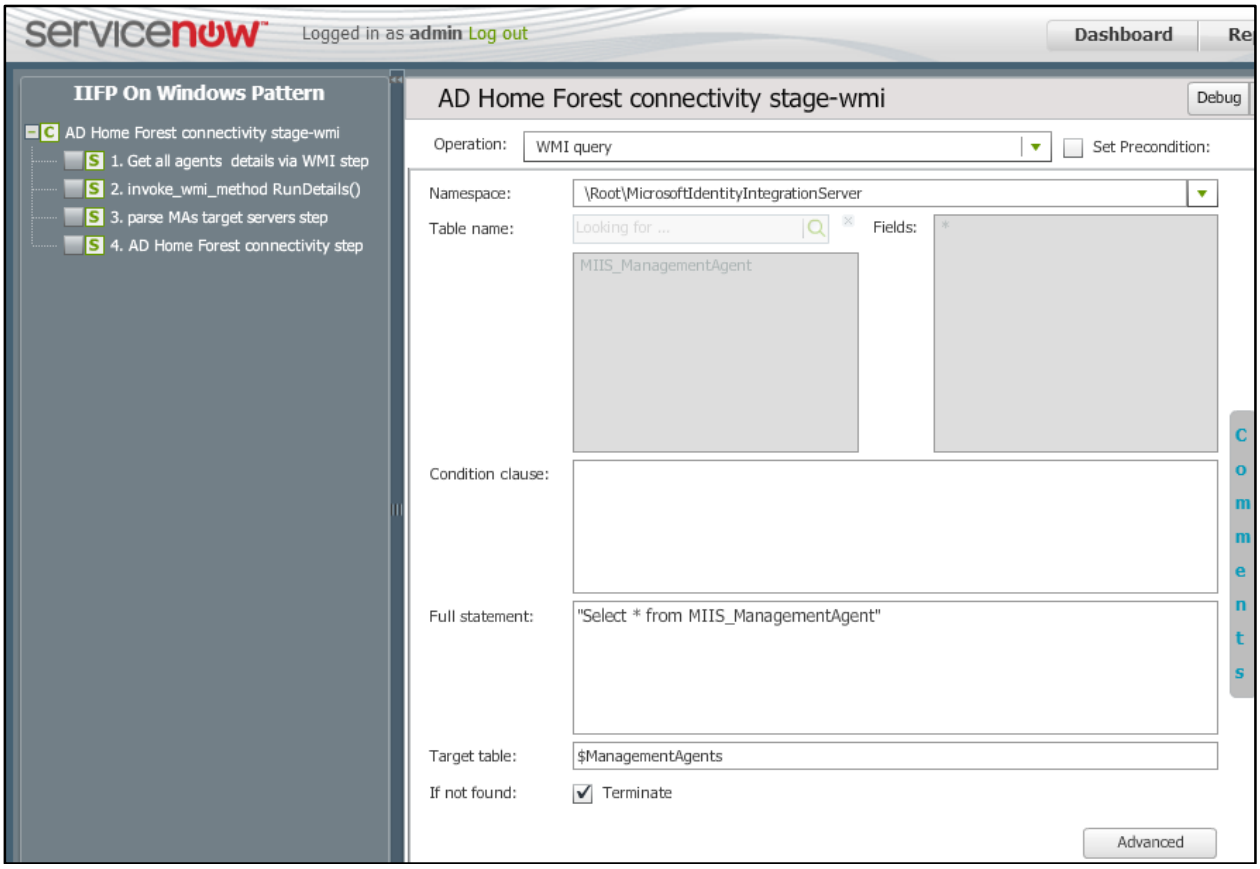
   Specify the **Condition clause**. As you type the condition clause, the query statement is written letter-by-letter in the **Full statement** line (a protected field).

   iv. Specify the **Target table** name.

   v. If discovery should terminate if no results are found, select (check) the **Terminate** checkbox.

Example

Figure 43: Letting ServiceWatch generate a WMI query



Example taken from:	
Hierarchy	Applications > Directory Services
CI Type	IIFP
Pattern	IIFP On Windows Pattern
Section	AD Home Forest connectivity stage-wmi
Step # and Name	1. Get all agents details via WMI step

Chapter 6: Business Service & Global Attributes

ServiceWatch provides global business service attributes (including the **bs_id** attribute that must not be changed or deleted) and supports user-defined global attributes (such as the **Port Monitor**). These attributes are listed in the **CI Attributes** tables of the **Global business service attributes** (Figure 44) and **Global attributes** (Figure 45) nodes in the **All CI Types** tree.

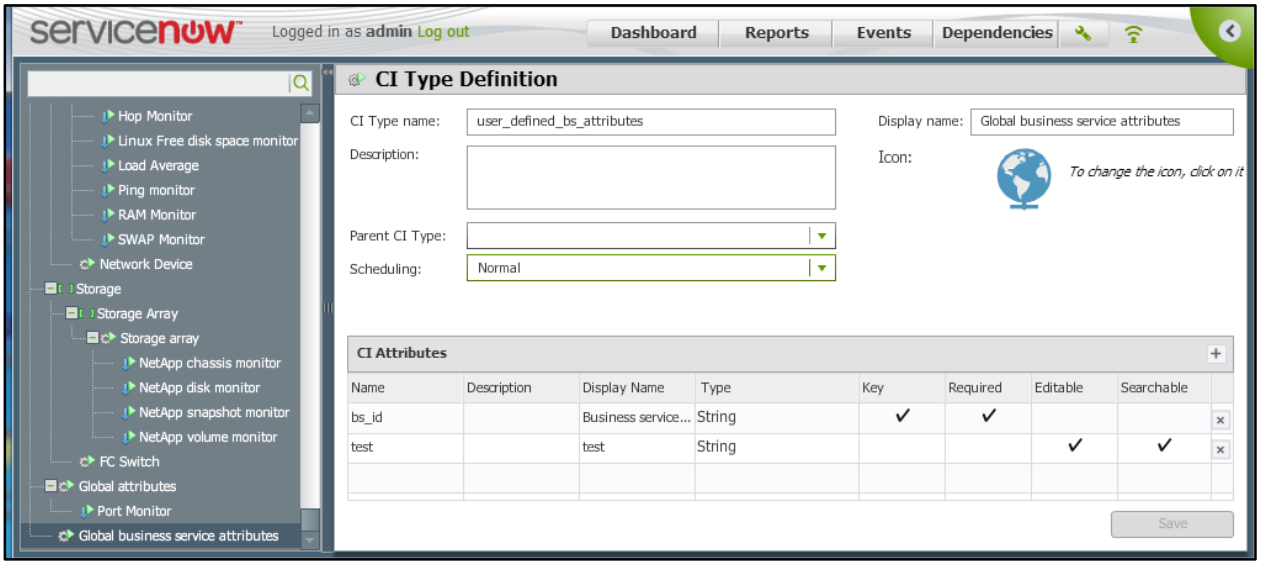
The administrator can define additional attributes that apply to every business service (see Global Business Service Attributes) and global numeric, Boolean, and string attributes that are automatically applied to every CI type (see Global Attributes).

Global Business Service Attributes

To define attributes that apply to every business service:

1. Click  in the Menu Bar. In the list of **Settings** options, click the **CI Types, Patterns & Monitors Definitions** link.
2. Select the **Global business service attributes** node at or near the bottom of the **All CI Types** tree. The screen illustrated in Figure 44 is displayed.

Figure 44: Global business service attributes screen



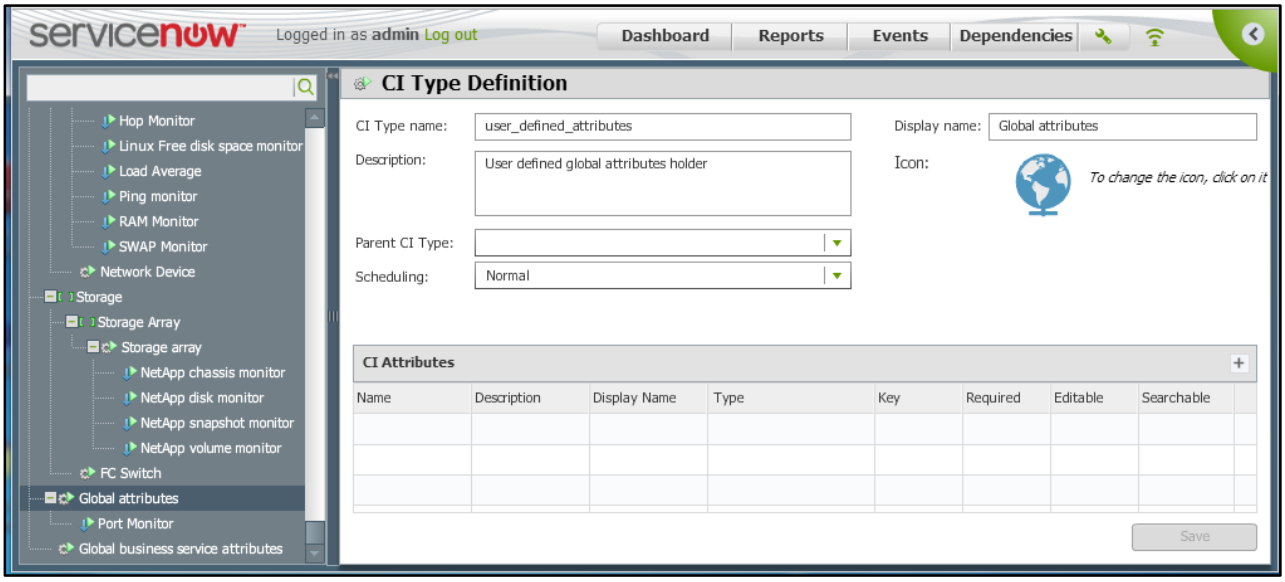
Caution: Do not modify the built-in global CI Attributes provided with the installation unless asked to do so by ServiceWatch customer support staff.

The procedure for defining new attributes and modifying or deleting existing attributes that apply to every business service is essentially identical to the procedure described in Defining CI Types on page 10. However, the administrator should not change the built-in **CI Attributes** shown above unless requested to do so by customer support staff.

Global Attributes

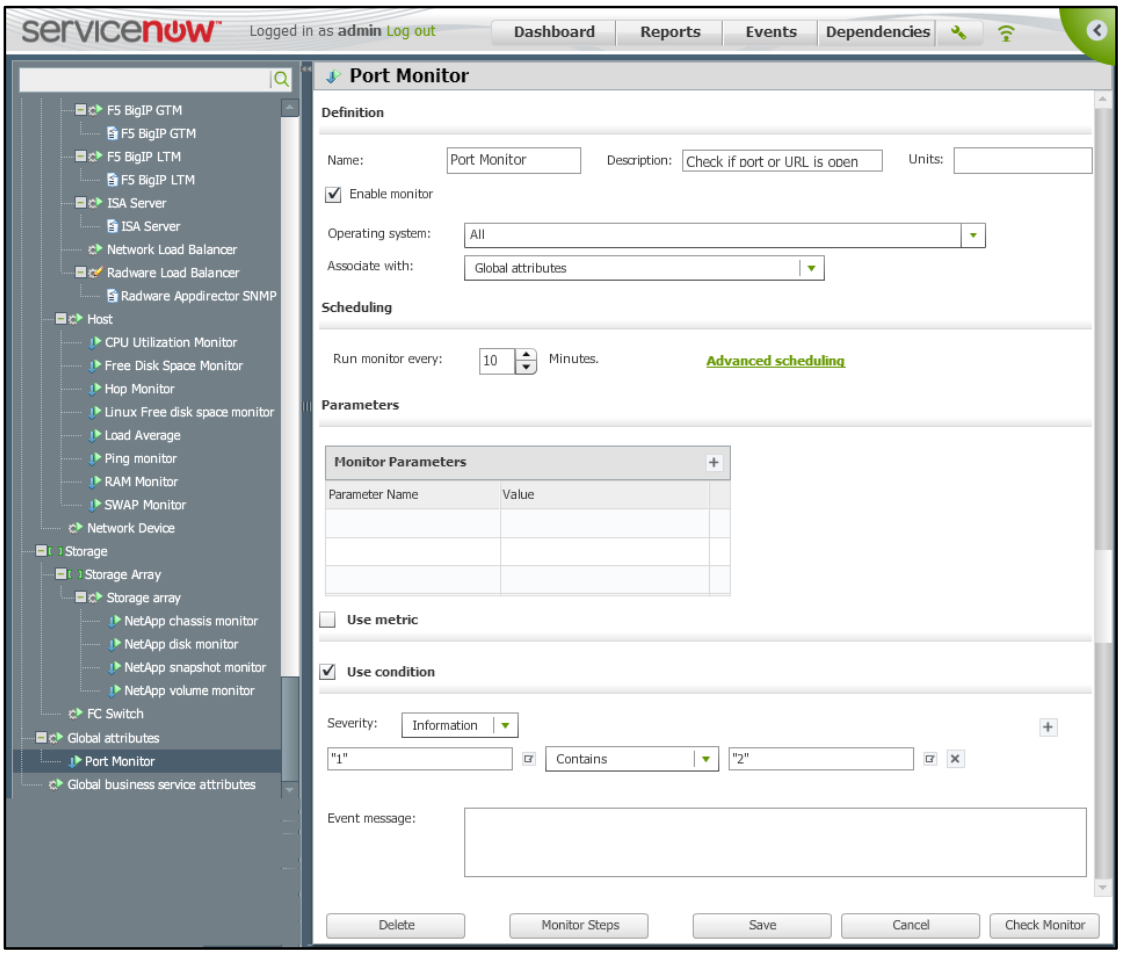
The **Global attributes** screen is used for defining global attributes that apply to *every* CI type. The procedure for creating a user-defined global attribute is essentially identical to the procedure for filling in the **CI Attributes** table as described in Defining CI Types on page 10. However, the administrator should not change the built-in global **CI Attributes** shown below unless requested to do so by ServiceWatch customer support staff.

Figure 45: Global Attributes screen



The Definition screen for the **Port Monitor** global attribute is shown in Figure 46:

Figure 46: Port Monitor definition screen



Appendix A: *Apache on Unix* pattern

This appendix describes the creation of an *Apache on Unix* pattern. The example demonstrates a complete pattern writing procedure and shows how to use the various steps and parsing strategies.

The purposes of Apache identification are:

- Identify and verify that the process listening on the entry point port is in fact an Apache web server process and not something else.
- Populate the Apache configuration item attributes with values. The attributes are:

  Label – the label that appears in the topology **Map** view

  Install directory – the directory where the Apache is installed

  Config file – Apache's configuration file (httpd.conf)

  Home directory – Apache's home directory

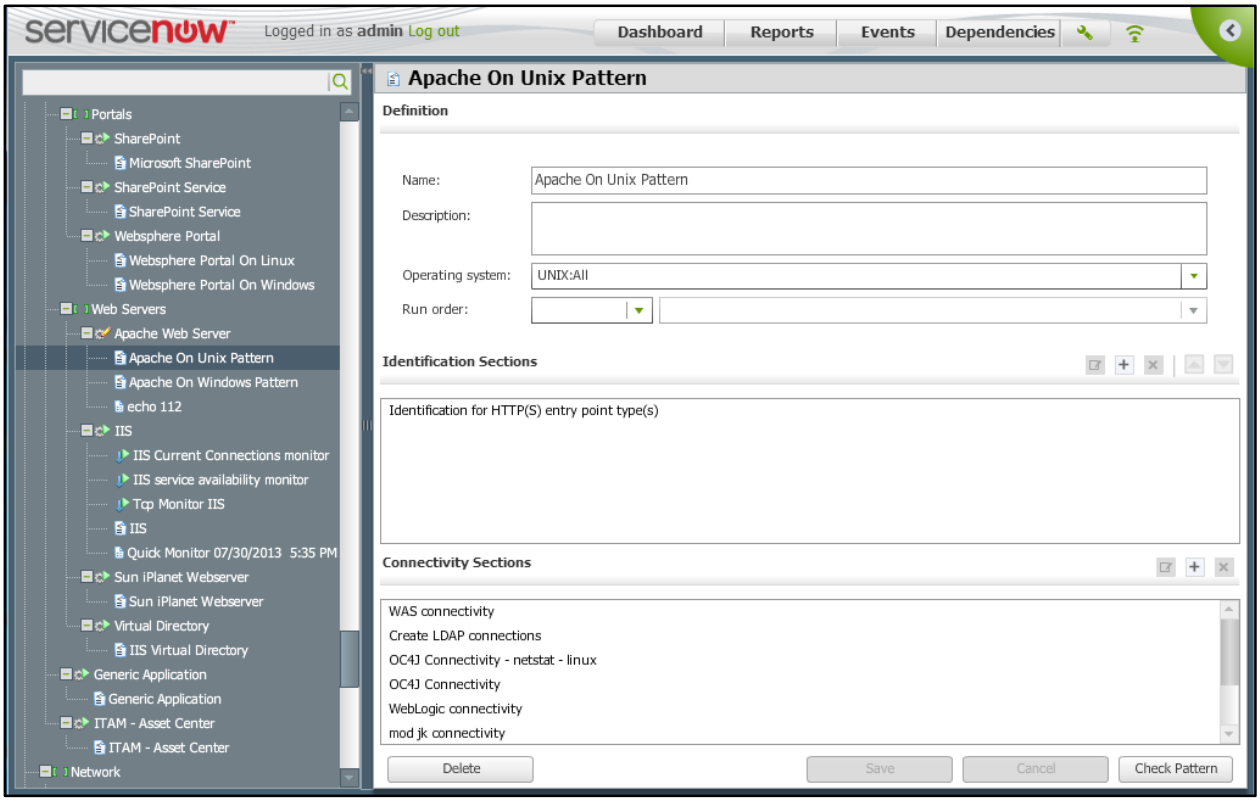
  Version – Apache version number

  Note: This attribute is determined but *not* populated.

Instructions for Creating a Pattern

1. To begin writing the pattern, choose the **Add new pattern** option from the right-click menu of the **Apache Web Server** node in the **CI Types** tree. This process is also described in Defining a General Pattern Definition on page 13.

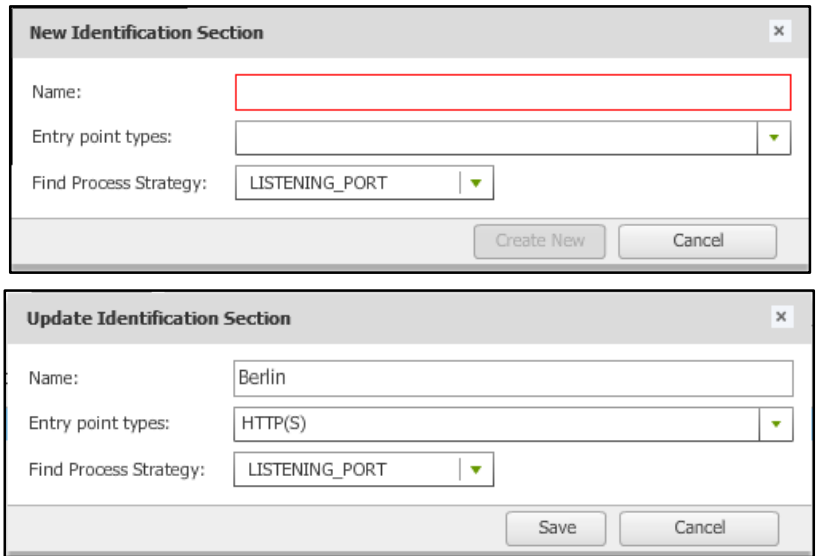
Figure 47: Apache On Unix Pattern window



Fill in the **Name**, **Description** (optional) and **Operating system**. The header will be filled in, character-by-character, as you type in the **Name**.

2. Click the plus icon ⊞ to display a **New Identification Section** dialog box or select an existing Identification Section and click the small checkbox on the left of the ⊞ to display its **Update Identification Section** dialog box.

Figure 48: New and Update Identification Section dialog boxes

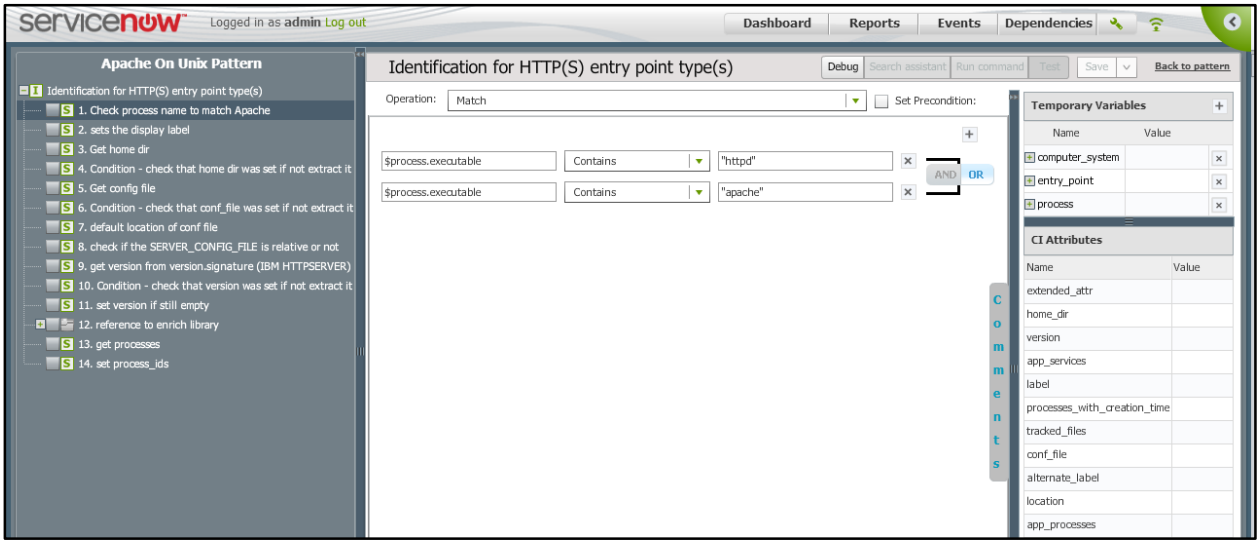


3. Assign or change the values for these fields.
   - ✓ **Name** of the port
   - ✓ For **Entry point types**, click its down arrow to select a value (in this example, HTTP/S).
   - ✓ For **Find Process Strategy**, click its down arrow and select LISTENING_PORT.

   and click the Create New or Save button.

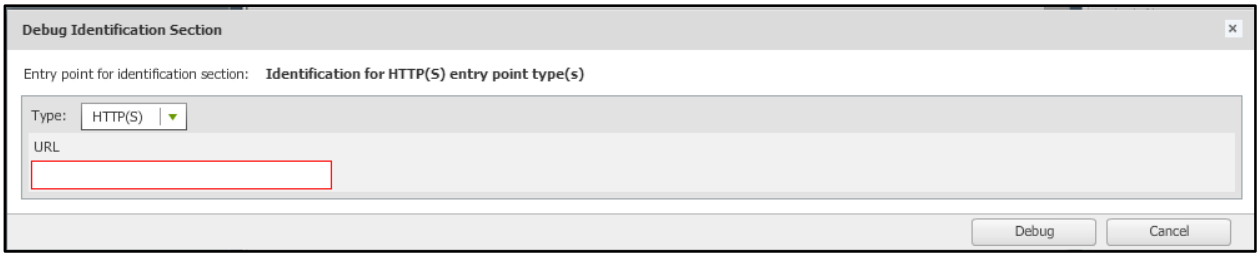
4. Double-click this text Identification for HTTP(S) entry point type(s) (not just anywhere on the row), or double-click the name of an existing Identification Section, to display its pattern editor.

Figure 49: Apache On Unix Pattern Editor



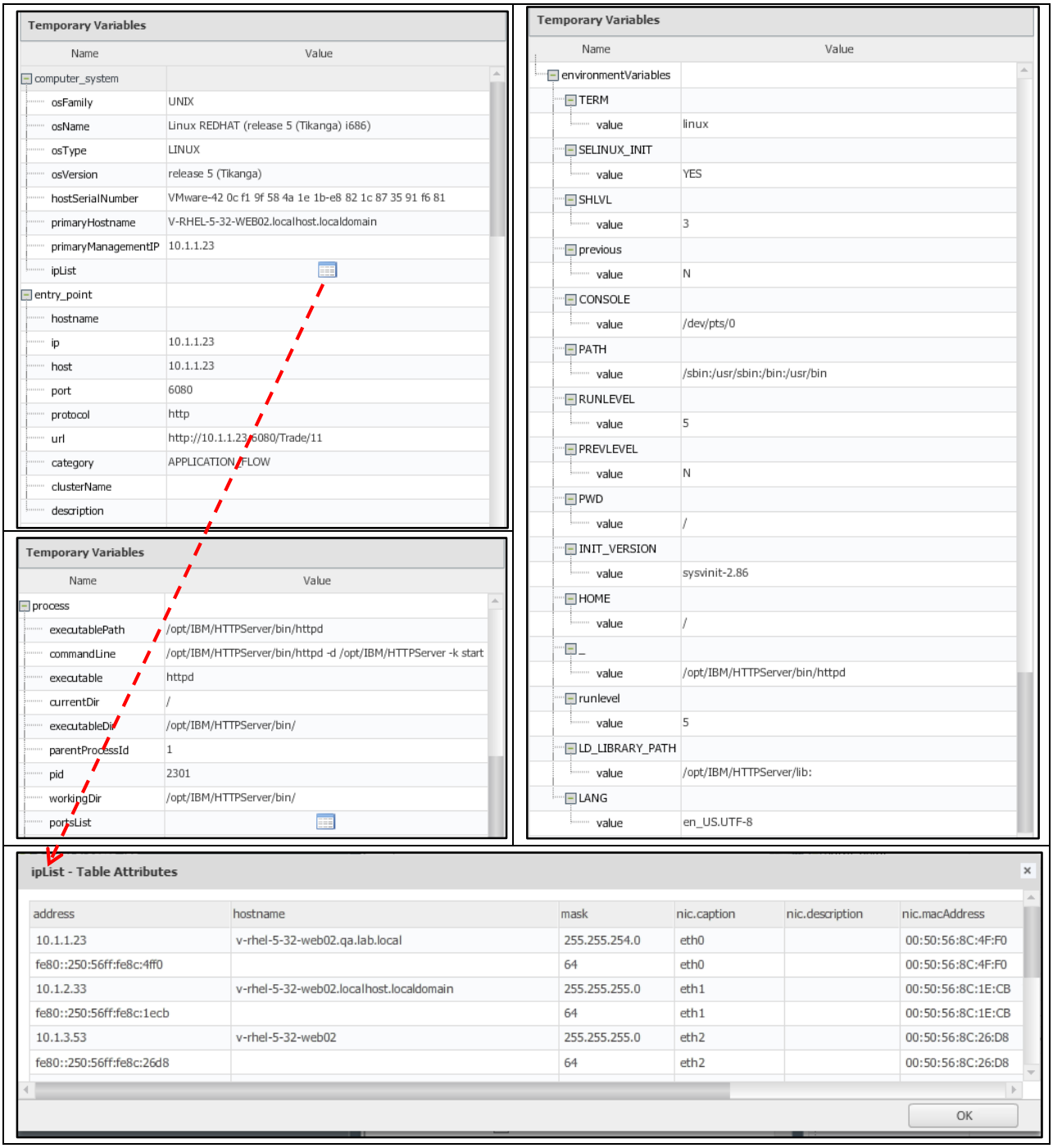
5. Start a debug session by clicking the Debug button. The **Debug Identification Section** dialog box is displayed.

Figure 50: Debug Identification Section dialog box



6. Insert the **URL** that is served by the Apache server you want to discover (in this example: http://10.1.1.23:6080/Trade/11) and click Debug. The debug session should populate the **Temporary Variables** pane with values for these variable types:
   - **computer_system** – the Apache's host information
   - **entry_point** – identified by the URL in this case
   - **process** – the Apache's process information
   - **environmentVariables** – that are displayed when a process variable is displayed.

Figure 51: Computer_system, Entry_point, Process, and EnvironmentVariables



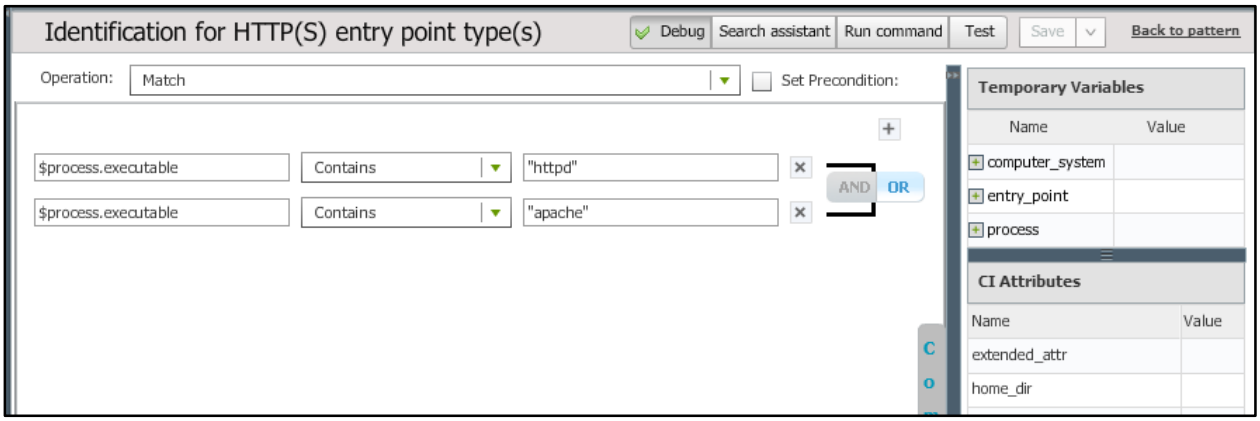
The green check in the Debug button indicates the process reached the entry point (URL). To see the variables (if any) in a table, click the table icon .

The **Apache on Unix Pattern** debug process contains the 14 steps shown in Figure 49**Error! Reference source not found.**. However, if at any time you need to return to the stage immediately before the first step (Figure 47), click the Back to pattern link.

Note: The number of steps in the debug process, and the step number of a step, varies by pattern type. Therefore, steps are identified by name and not by step number. Clicking the [S] in a step collects data for that step by running the previous steps. To run the selected step, click the Test button. Because the [S] in the last step only runs the previous steps, the only way to run the last step is to click its Test button.

7. Perform the **Check process name to match Apache** step to determine whether the process listening on the entry point's port is in fact Apache. Use the Match operation (see Figure 52). Verify that the query specification is **$process.executable Contains** the strings **httpd** *or* **apache** and that the OR half of the AND OR toggle is active (not grayed-out).

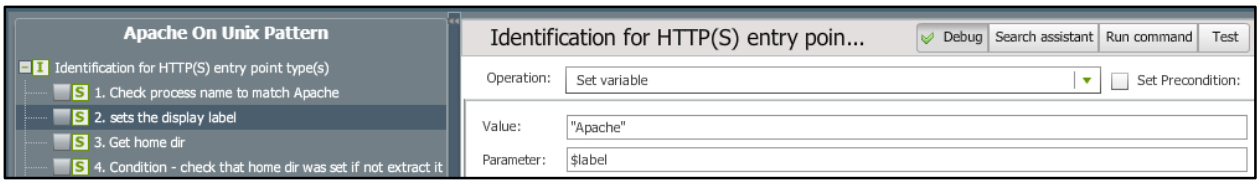
Figure 52: Match Operation in the Identification for HTTP(S) entry point type(s) dialog box



If you click the Test button, you should get the message Debug Results: No changes were made during this test.

The **Set variable** Operation populates the *label* attribute. In the **sets the display label** step, set **Value** to "**Apache"** (enclosed in quote marks) and **Parameter** to **$label**.

Figure 53: In the sets the display label step, the Set variable Operation populates the label attribute

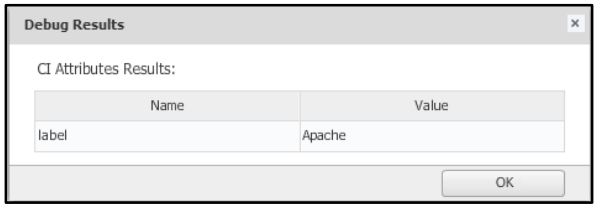


Click Test *or* click [S] in the *next* step to Debug through *this* step.

Note: To test/debug an *existing* pattern (instead of creating a new pattern), select the last step through which you want the process to run and click its Test button. The process should run until the end of the selected step unless an error occurs in an earlier step.

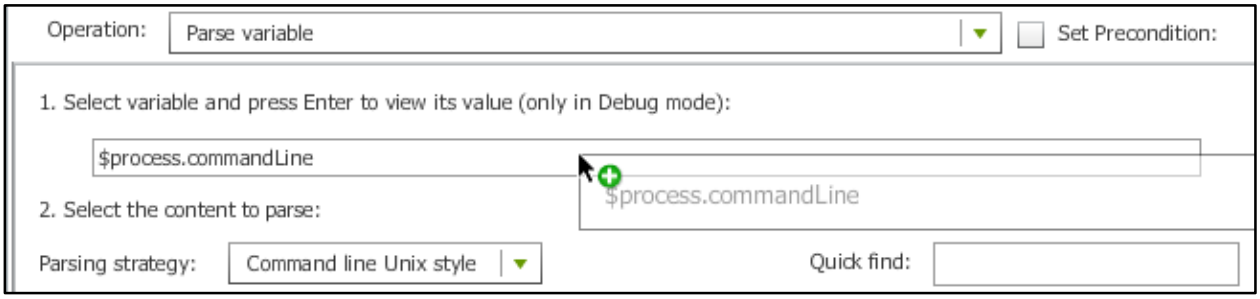
Click OK when you receive the message:

Figure 54: Debug Results Message Box



8. The **Get home dir** step populates the *home_dir* attribute.
   a. Select Parse variable from the **Operation** drop-down list to extract the value after the **–d** in the content box.
   b. Drag **commandL**ine from the **process** area **Name** column to move **$process.commandLine** to the textbox under **1. Select variable and press** Enter ... . To do this, release the mouse button while  is visible. The process command line value (**/opt/IBM/HTTPServer/bin/httpd** in this case) is displayed in the large content box.

Figure 55: Dragging the command line from the process area of the Temporary Variables pane

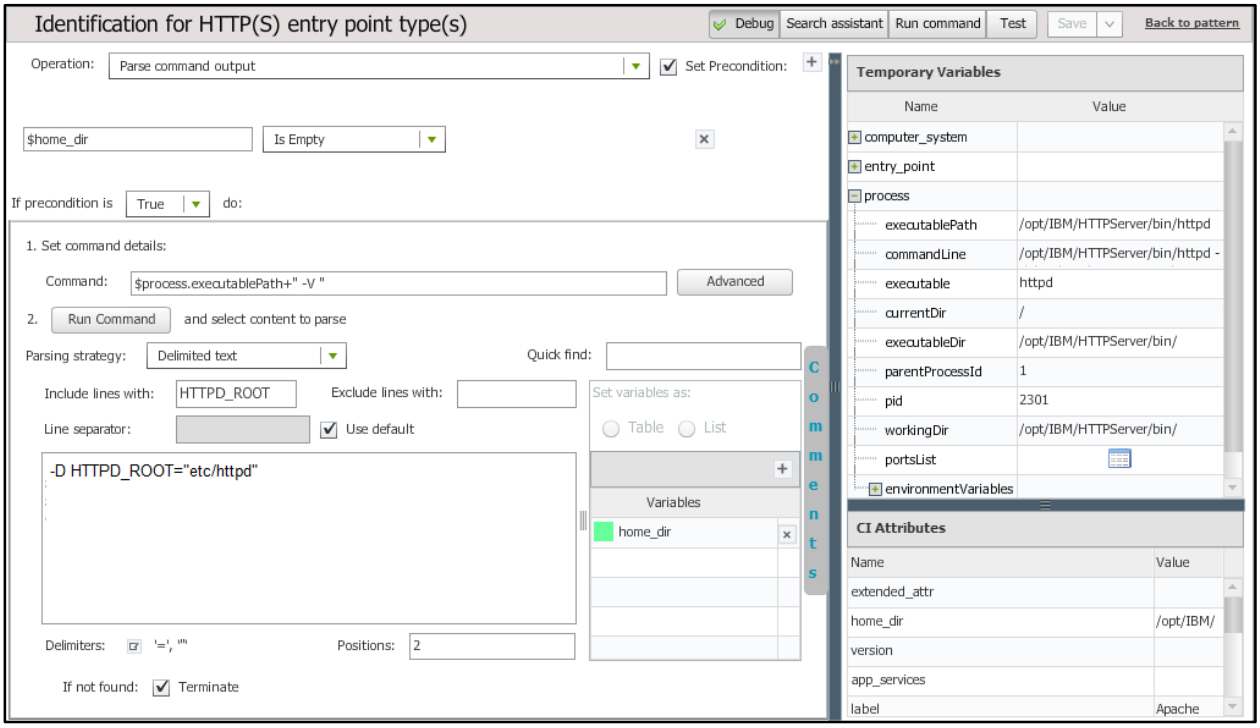


   c. Click the Test button.

Note: If CI Attribute **home_dir** was retrieved, skip the next step. In this case, it was not retrieved, probably because **–d** was not specified, so you *do* need to run the next step.

9. The next step, **Condition – check that home dir was set if not extract it from httpd –V**, parses the $process.executablePath+" -V " command to obtain the home_dir value.
   a. Use the **Parse** command output **Operation** with a **$home_dir Is Empty** condition.
   b. Set the **Command** textbox to $process.executablePath+" -V "
   c. Click the Run Command button.
   d. Select **Delimited text** from the **Parsing strategy** drop-down list.
   e. Set **Include lines with** to **HTTPD_ROOT**
   f. Set **Delimiters** to **'='**,**'"'**, that is, to the equal sign and double quote mark, and **Positions** to **2**

Figure 56: In the check home dir step, the Parse command output populates the home_dir attribute



g. Click the Test button.

h. Click the OK button in the **Debug Results** dialog box when you receive the message

Figure 57: Debug Results dialog box

10. Click the **Get config file** step. The **home_dir** value (**/opt/IBM/HTTPS** in this case) is assigned in the **CI Attributes** pane.

    a. Select **Parse variable** from the **Operation** drop-down list.

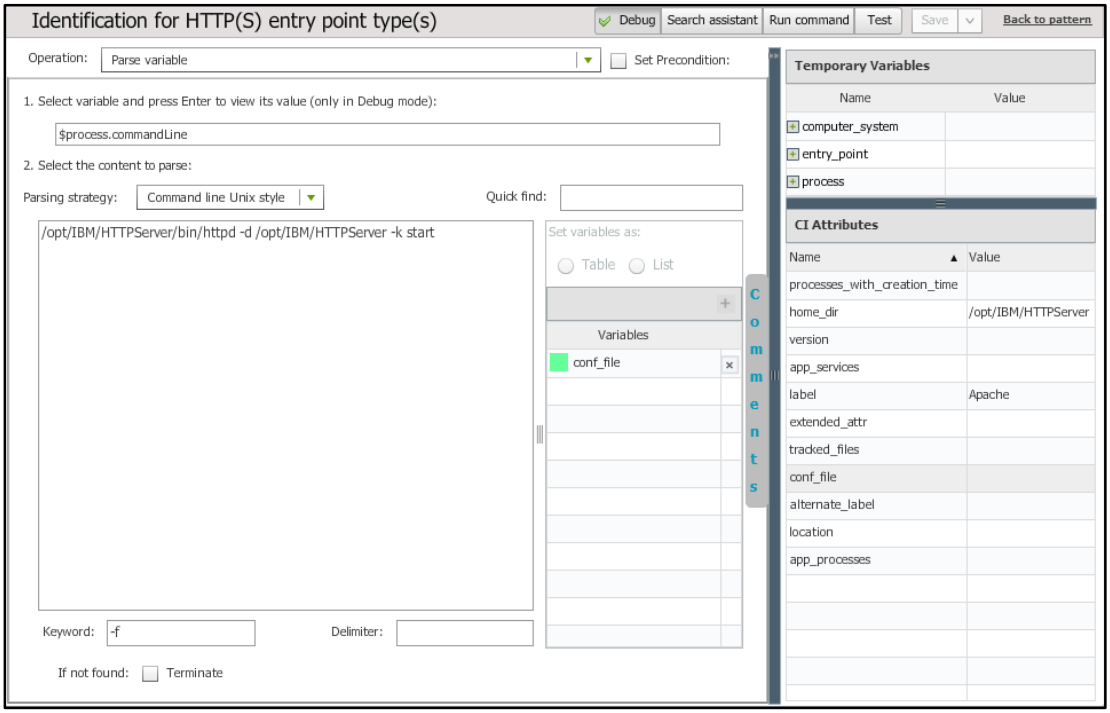
    b. Drag the **command** line from the **process** area of the **Temporary Variables** pane to move the **$process.commandLine** attribute to the textbox under **1. Select variable and press** ...

    To do this, release the mouse button while the  is visible. See Figure 55. The process command line value (**/opt/IBM/HTTPServer/bin/httpd** in this case) is displayed in the content window.

    c. Select **Command line Unix style** from the **Parsing strategy** drop-down list.

    d. Click the Test button.

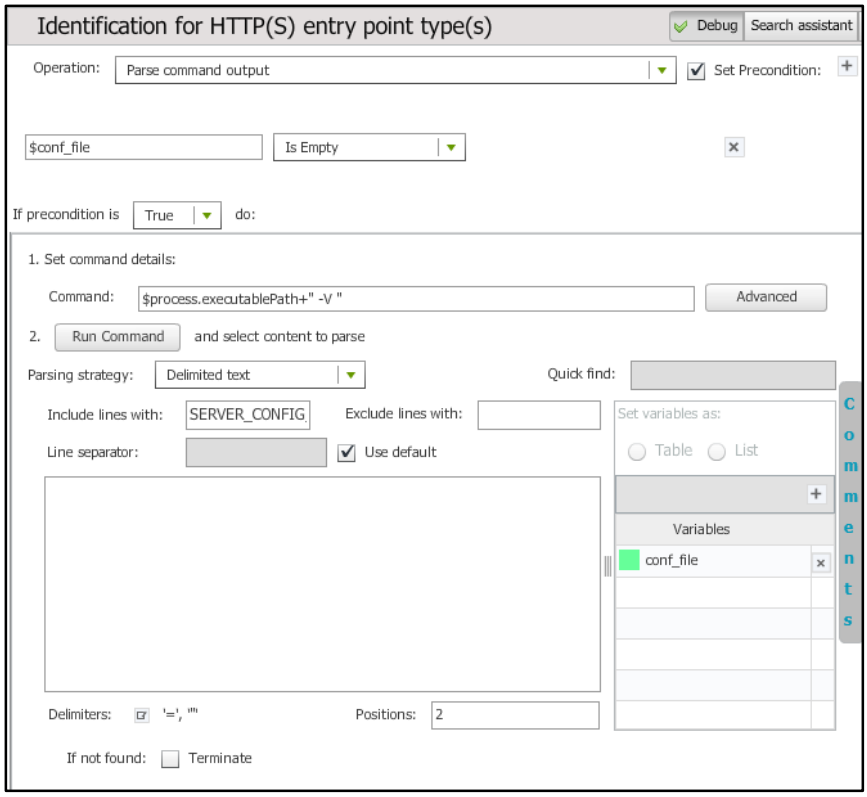
Figure 58: In the Get conf_file step, the Parse variable Operation populates the conf_file attribute



Note: If CI Attribute **conf_file** was retrieved, skip the next step. In this case, it was not retrieved, probably because **-f** was not specified, so you *do* need to run the next step.

11. The next step, Condition – check that conf_file was set if not extract it from httpd –V, parses the $process.executablePath+" -V " command to obtain the conf_file value.
    a. Use the **Parse command output** Operation with a **$conf_file Is Empty** condition.
    b. Set the **Command** textbox to **$process.executablePath+" -V "**
    c. Click the 

button.
    d. Select **Delimited text** from the **Parsing strategy** drop-down list.
    e. Set **Include lines with** to **SERVER_CONFIG_FILE**
    f. Set **Delimiters** to **'=','""'**, that is, to the equal sign and double quote mark, and **Positions** to **2**



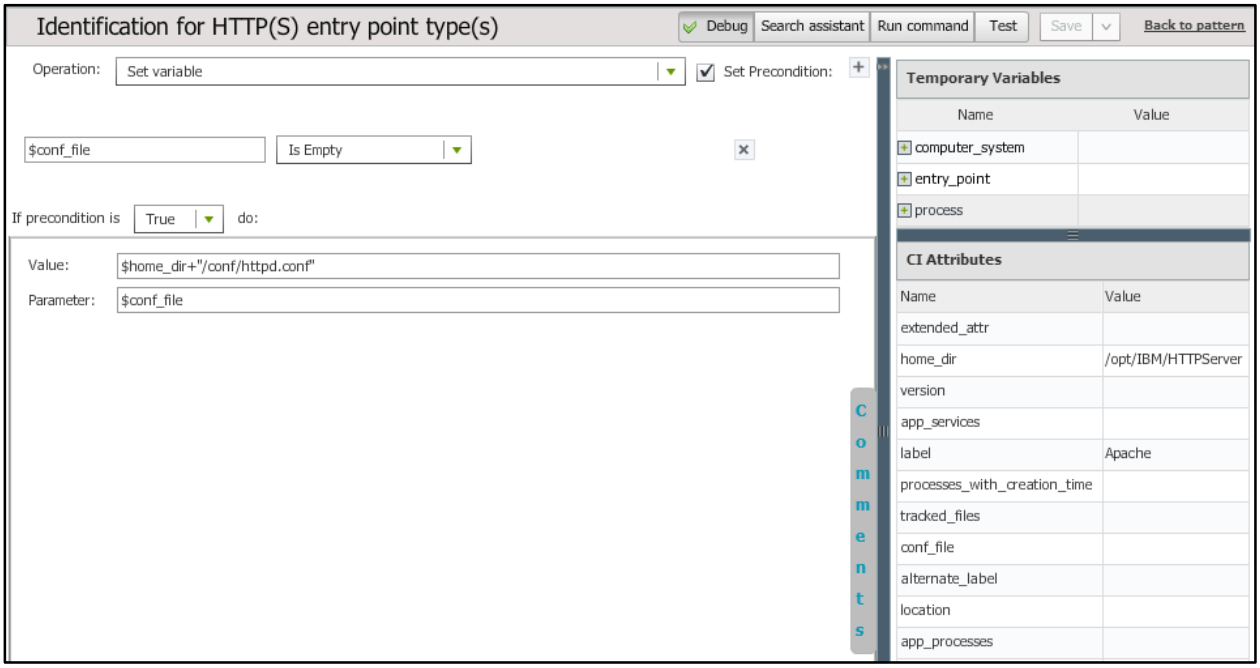
g. Click the Test button.

h. Click **OK** when you receive the message

Figure 59: Debug Results dialog box

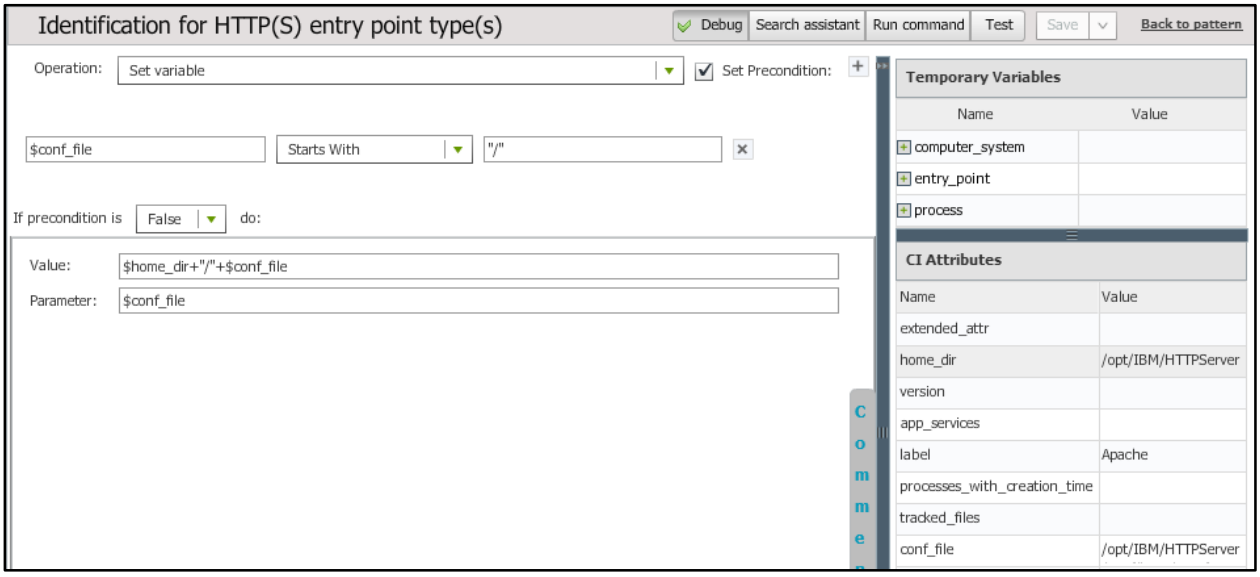
12. Because the value of **conf_file** was found by the previous step, you do *not* need to run this **default location of conf file** step. However, if the **conf_file** value was not found, you would run this step by specifying the values shown in Figure 60 and clicking the Test button.

Figure 60: The default location of conf_file step populates an *empty* conf_file attribute



13. The **check if the SERVER_CONFIG_FILE is relative or not** step concatenates the **home_dir** and **conf_file** values using a forward slash **/**. Run this step by specifying the values shown in Figure 61 and clicking the Test button.

Figure 61: Concatenating the home_dir and conf_file in the SERVER_CONFIG_FILE step



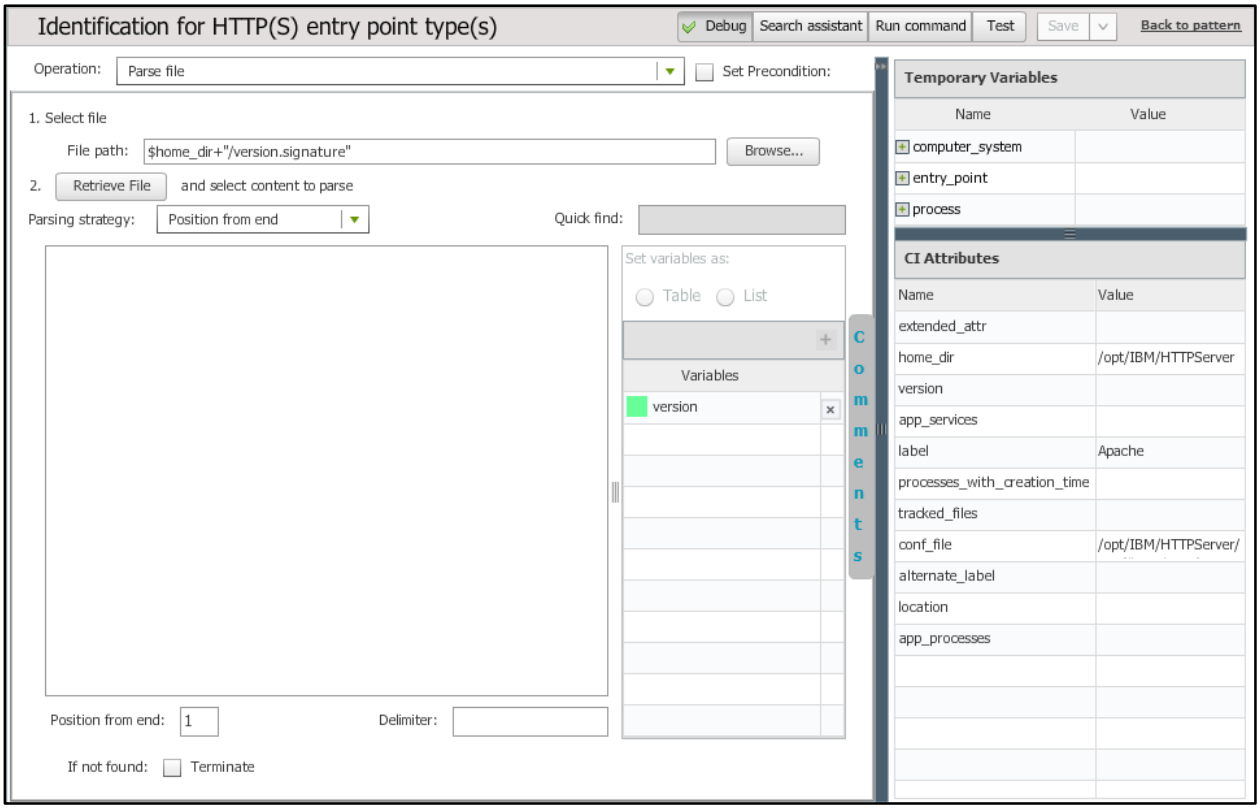
14. The **get version from version.signature (IBM HTTPSERVER)** step populates the version attribute. To do this we parse the **version.signature** file that is located in the Apache's home directory. See Figure 62.
    a. Select **Parse file** from the **Operation** drop-down list.
    b. Set **File path** to a concatenation of **$home_dir** and the string **"/version.signature"**

c. Click the Retrieve File button to view the file content.

d. If the file is found, click the Test button.

e. Otherwise, go to the next step to extract the version from **httpd**.

Figure 62: Retrieving the version number using the get version from version.signature step



15. The version was not extracted by the previous step, so it will be set by the **Condition – check that version was set if not extract it from httpd** step. See Figure 63.

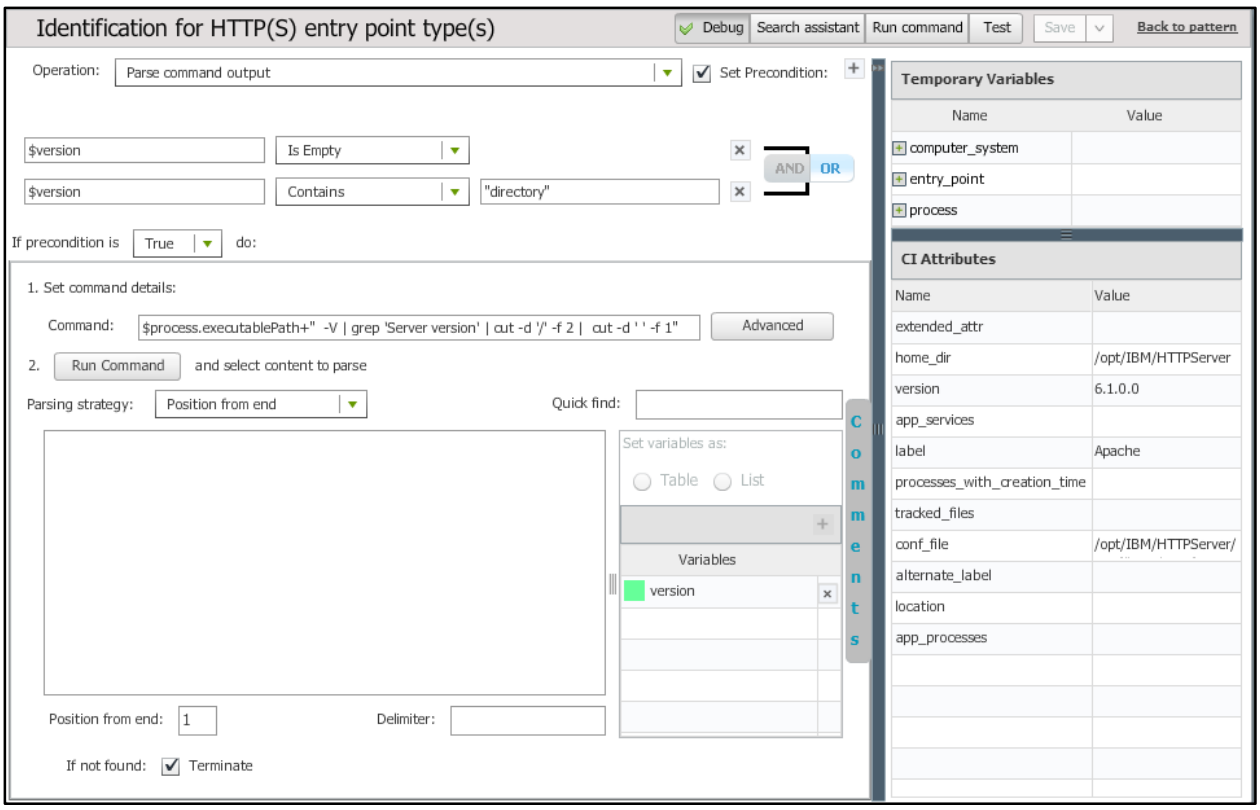
a. Select **Parse command output** from the **Operation** drop-down list.

b. **Set Precondition** to **$version Is Empty** AND **$version Contains "directory"**.

c. Set **If precondition is** to **True**.

d. In the **Command** textbox, specify
$process.executablePath+" -V | grep 'Server version' | cut -d '/' -f 2 | cut -d ' ' -f 1"

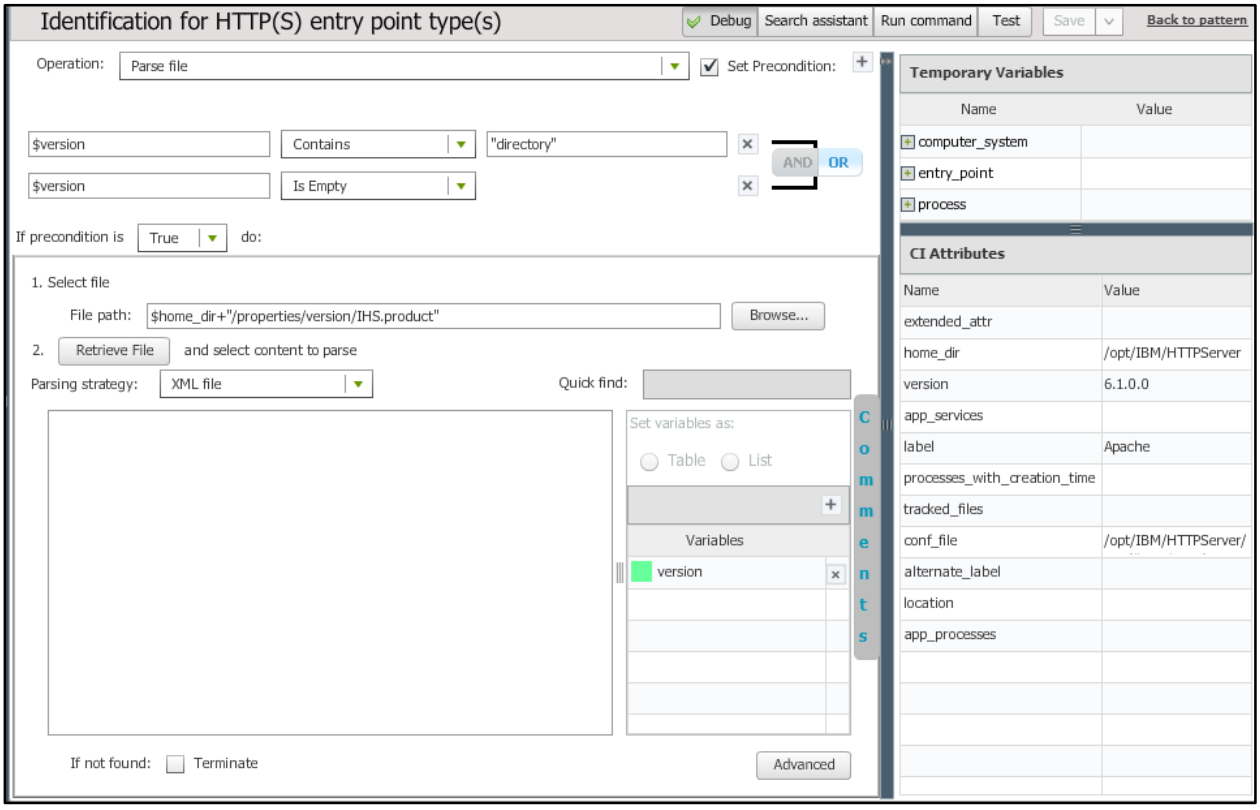
e. Click the Run Command button. The version number should appear in the **Content** box.



16. Click the Test button. When the **version Value** is displayed, click **OK**.

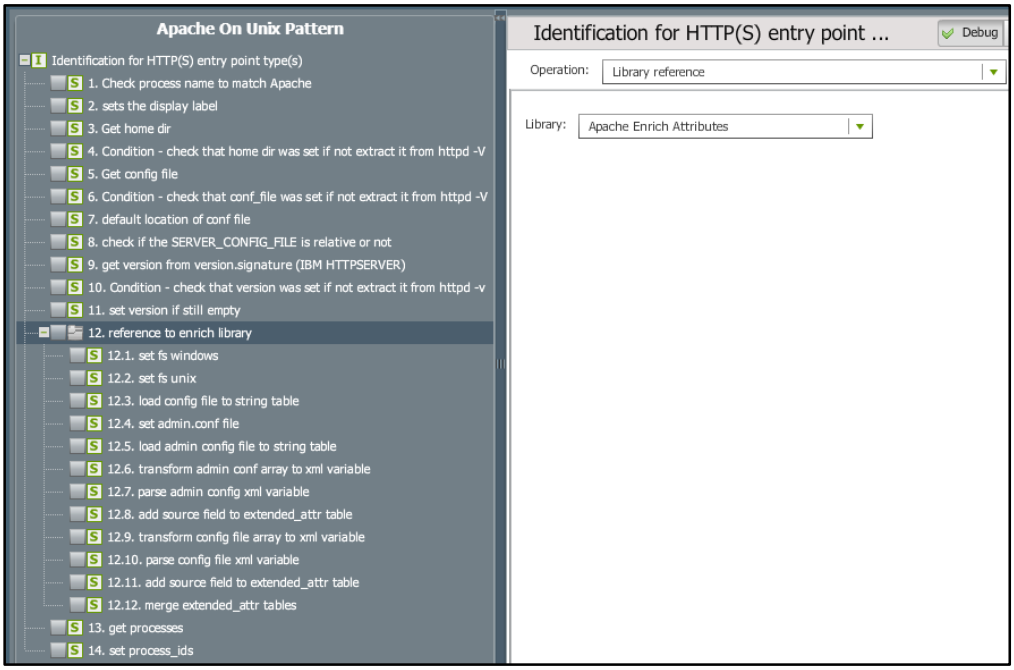
Note: The version number is *not* moved into the **CI Attributes** pane because confirmation of its value in the Content box and the **Debug Results** dialog box is sufficient.

Figure 63: Set version if still empty step

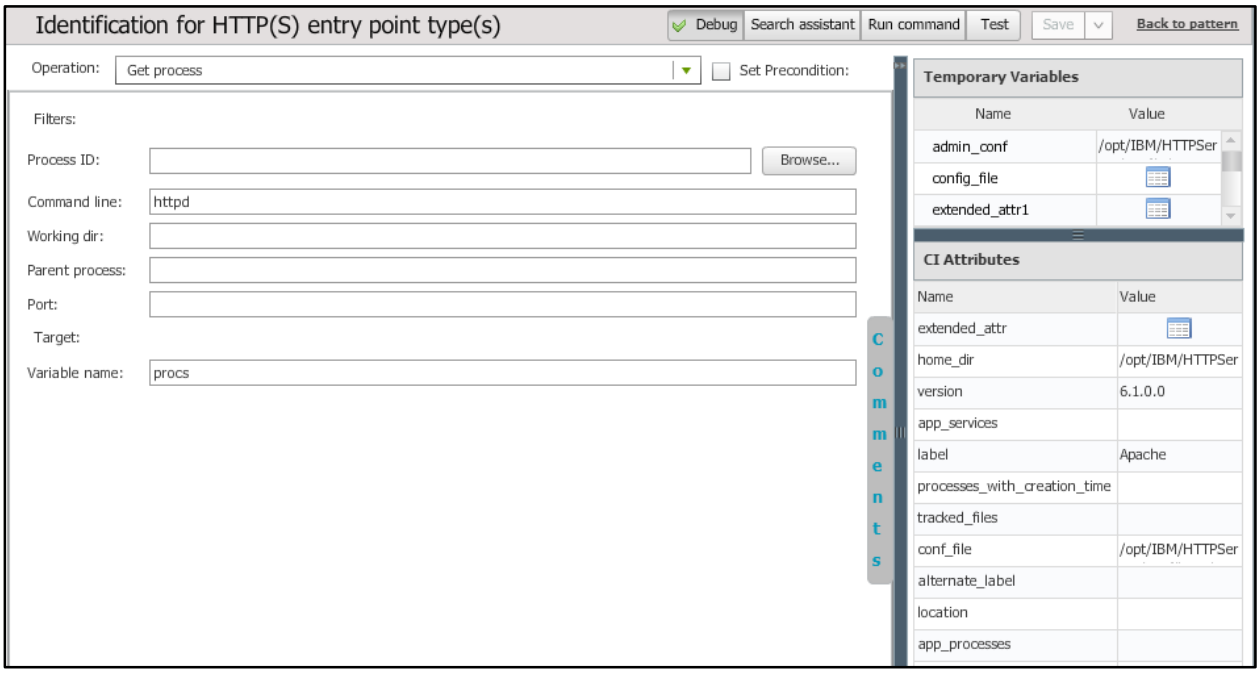


Note: You have successfully identified the Apache server and populated its various attributes (except the version attribute that is intentionally not populated).

17. Run the **Library reference Operation** that invokes the 12 sub-steps of the **Apache Enrich Attributes** library.

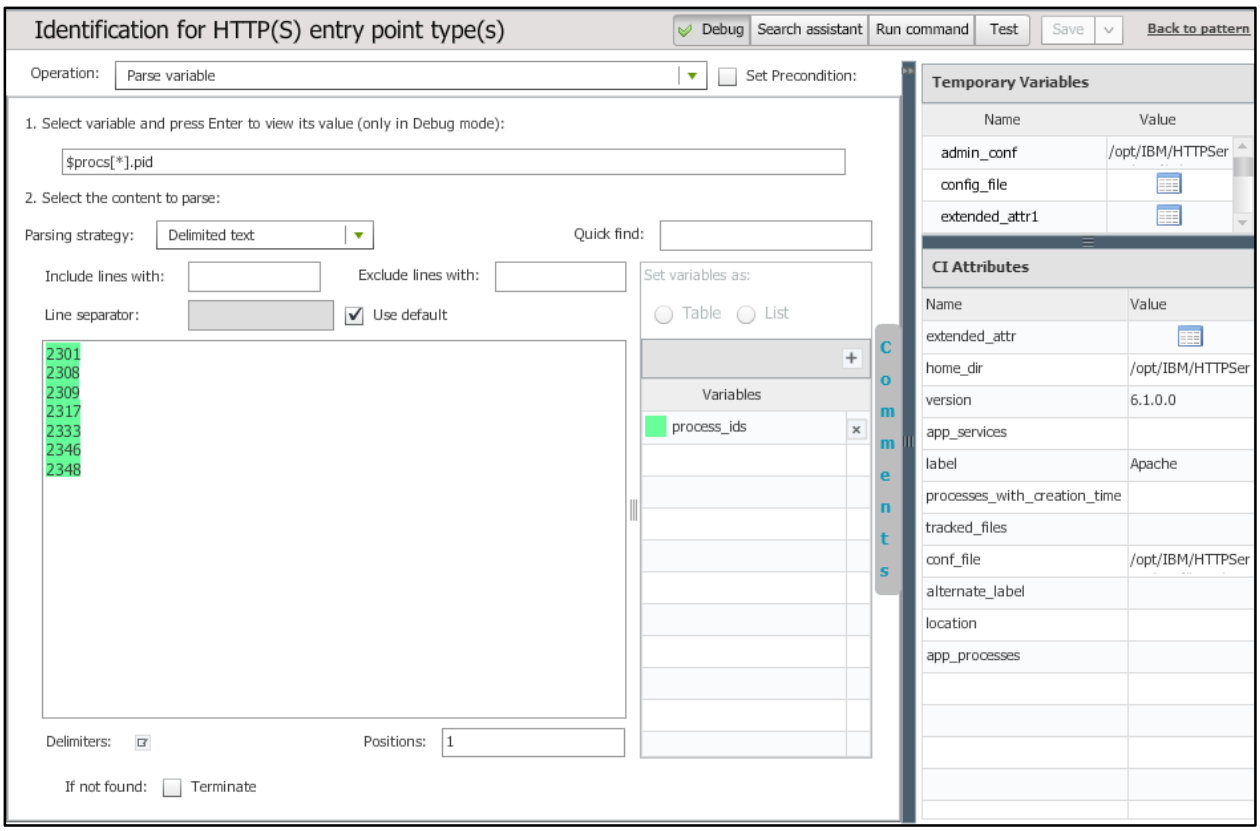


18. Perform the **Get process** step.



19. Click the Test button. The **procs – Table Attributes** are displayed.

20. Perform the set process_ids step. The process IDs are automatically displayed in the content window.


Identification for HTTP(S) entry point type(s)
Debug
Search assistant
Run command
Test
Save
Back to pattern
Operation:
Parse variable
Set Precondition:
1. Select variable and press Enter to view its value (only in Debug mode):
$procs[*].pid
2. Select the content to parse:
Parsing strategy:
Delimited text
Quick find:
Include lines with:
Exclude lines with:
Line separator:
Use default
2301
2308
2309
2317
2333
2346
2348
Delimiters:
Positions:
1
If not found:
Terminate
Set variables as:
Table
List
Variables
process_ids
Comments
Temporary Variables
Name
Value
admin_conf
/opt/IBM/HTTPSer
config_file
extended_attr1
CI Attributes
Name
Value
extended_attr
home_dir
/opt/IBM/HTTPSer
version
6.1.0.0
app_services
label
Apache
processes_with_creation_time
tracked_files
conf_file
/opt/IBM/HTTPSer
alternate_label
location
app_processes

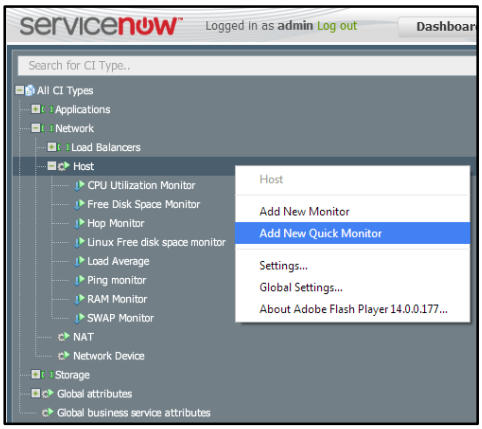
Appendix B: Creating a *Simulation* monitor

This appendix describes the creation of a monitor that generates random numbers from 0 to 100 to simulate the percent usage of a CPU or storage device. The example demonstrates a complete monitor creation procedure and shows how to use monitor steps.

Instructions for Creating a Monitor

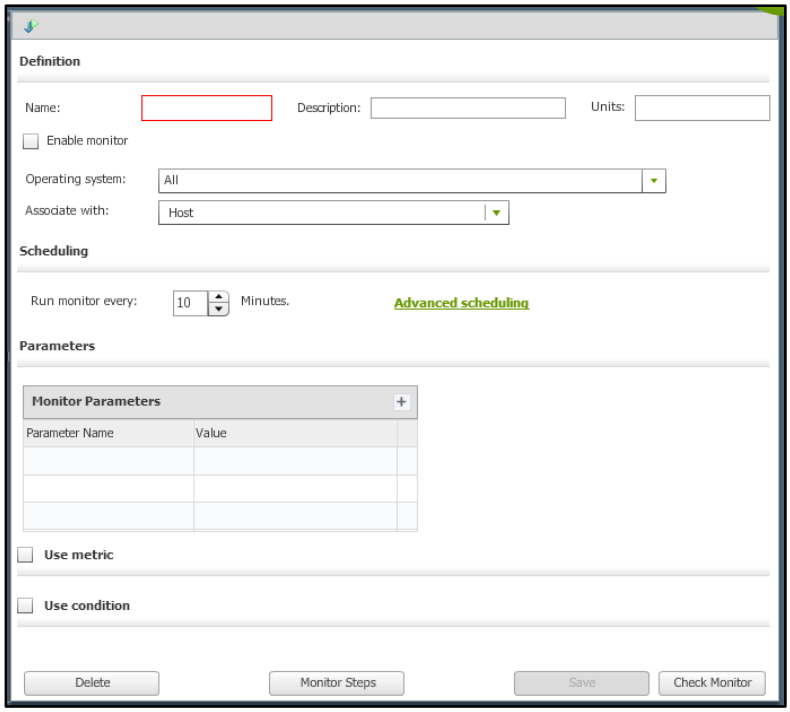
1. In the **Settings Menu**, click the **CI Types, Patterns & Monitors Definitions** link and right-click the **CI Types** node at the level you want to monitor. In this example, we right-click a Host node because we want to monitor all of the CI Types under that Host.

Figure 64: Right-click menu with Add New Monitor Option



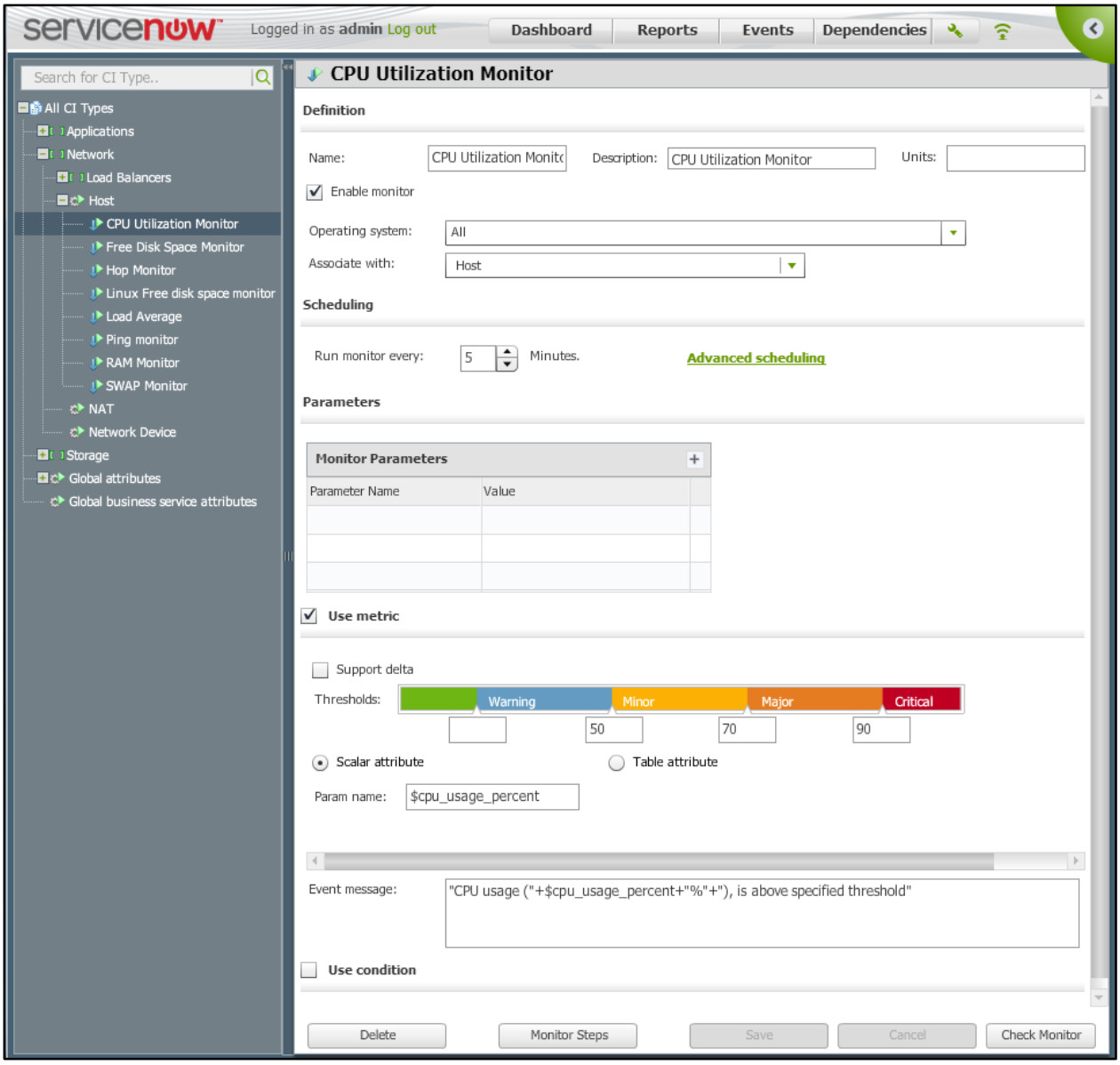
2. Click the **Add New Monitor** option. An "empty" **New Monitor** screen is displayed.

Figure 65: New Monitor screen



3. Fill in the **Name** and (optional) **Description**. The header will be filled in, character-by-character, as you type in the **Name**.

Figure 66: CPU Utilization monitor screen (*after* the monitor is defined)



4. If you want this monitor to be enabled automatically when its Business Service is activated, select (check) the **Enable monitor** checkbox.

5. Select a value from the **Operating system** drop-down list. If you select **UNIX**, you must select one or more UNIX types. If you select all of the UNIX types, the value in this field changes to **UNIX:All**

6. Select a value from the **Associate with** drop-down list. This field enables you to specify whether the monitor applies to the entire CI or only to only some more specific aspect of the CI.

7. In the **Run monitor every** field, specify (or use the up and down arrows to specify) the interval in minutes at which the monitor will run.

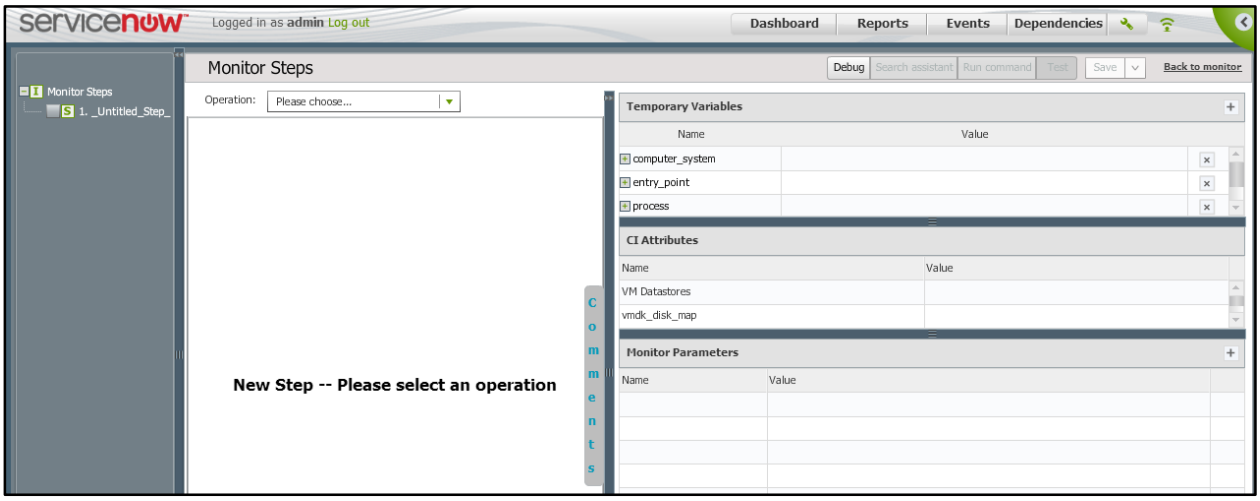
8. Click ⊞ in the **Monitor Parameters** table to specify one or more parameter **Name**s that will trigger the monitor when its specified **Value** is matched. In this example, no such parameters have been specified.
9. Select the **Use metric** checkbox to define ranges that will cause the current **Event Severity** designation (Critical, Major, Minor, Warning) to be stored in a Parameter or Table attribute. Select the **Support delta** checkbox if the defined ranges represent the amount by which the previously monitored value has changed.
10. Select the **Scalar attribute** or **Table attribute** radio button to specify whether the **Event Severity** value is stored in the **Param name** parameter or in a in a table identified by **Table name**, **Key column** and **Value column**.
11. Use the **Event message** textbox to specify the message that is generated each time the monitor runs.
12. Select the **Use condition** checkbox if the monitor is used only when a specified condition is satisfied.
13. Click Monitor Steps to specify one or more steps that are performed when the monitor runs. See Creating Monitor Steps below.
14. After the steps have been defined, click Save to save the monitor definition.

Creating Monitor Steps

The Monitor steps are performed each time the monitor runs.

1. Click the Monitor Steps button (see step 13 above). An “empty” first step is displayed.

Figure 67: Monitor untitled Step 1

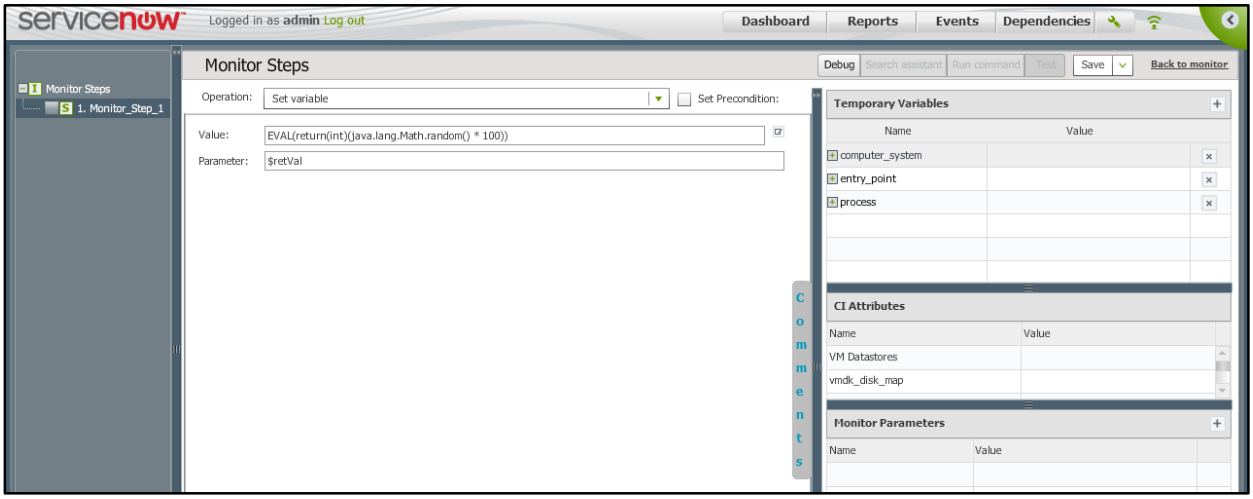


2. Select **Set variable** from the **Operation** drop-down list.
3. If a condition must be satisfied for this operation, select the **Set Precondition** checkbox. Specify the condition and click the ⊞ icon for each additional condition. In this example, there are no preconditions.

4. Type the string **`EVAL(return(int)(java.lang.Math.random() * 100))`** in the **Value** textbox and the string **`$retVal`** in the **Parameter** textbox.

5. Double-click the string **_Untitled_Step_** and overwrite it with **Monitor_Step_1**. The result should look similar to Figure 68.

Figure 68: Monitor Step 1 completed



6. Click the **Return to monitor** link and click 

to see the result of running the monitor immediately.