Auto Discovery

Overview

The Auto Discovery tool allows administrators to automatically coordinate network resources with GroundWork Monitor's configuration database.

CONTENTS

RELATED RESOURCES

- NeDi

WAS THIS PAGE HELPFUL?

- Leave Feedback

About Auto Discovery

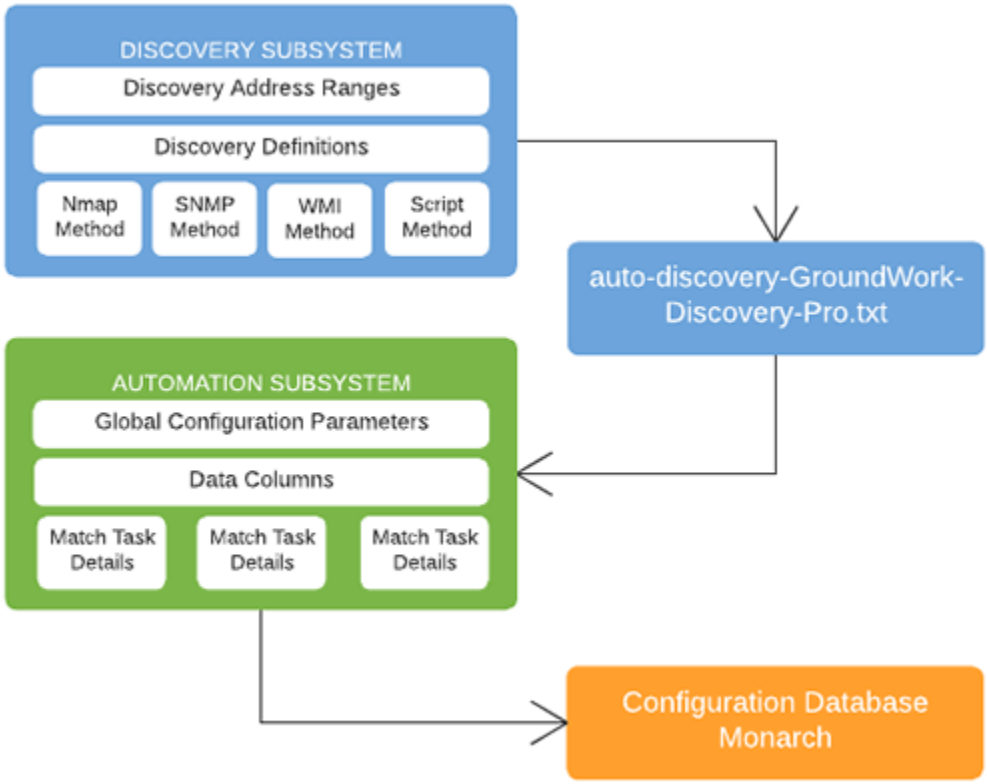
The Auto Discovery Subsystems

This tool allows administrators to automatically coordinate network resources with GroundWork Monitor's configuration database, thereby relieving administrators from having to manage all of their host and service definitions manually. In a typical scenario, administrators would use this tool to probe network addresses for host devices that meet specific criteria. Once the probes have completed, the devices are matched with the most appropriate host and service profiles, and the administrator is given the chance to review the resulting data before it is committed to GroundWork Monitor's configuration database.

There are two major subsystems in the Auto Discovery tool that are used to perform the process described below. First, a **Discovery** subsystem is responsible for performing the device-specific probes, and then generates output data from these probes. Separately, an **Automation** subsystem is responsible for parsing through the discovery data in order to determine the appropriate host and service profiles, and is also responsible for updating the GroundWork configuration database with the final results.

Network discovery operations such as the one described here require both of these components in order to perform as expected. However, it's important to recognize that the Automation subsystem is not exclusively dependent on the Discovery component to provide its input data, and in fact, the Automation component can use almost any kind of structured text for the purpose of synchronizing the configuration database. For example, administrators can use the Automation subsystem to synchronize GroundWork Monitor's configuration database with an input file that originated from a spreadsheet or a network management system, without ever probing the network at all. In this regard, the Discovery and Automation components are relatively independent, and perform different tasks, although both of them play important roles, and they are designed to work together in common usage scenarios.

Figure: Auto Discovery Architecture



Part I: The Discovery Subsystem

1.0 About The Discovery Subsystem

In practice, the discovery process is relatively straightforward, and usually only requires an administrator to answer a few confirmation prompts. However, there are a large number of options and variables that must be defined before this process can be started.

1.1 Discovery Definitions

In GroundWork Monitor, discovery definitions are used to store the configuration options for a particular discovery process. Once a discovery definition has been created, administrators can simply select the most appropriate discovery definition for the task at hand and immediately begin the associated discovery process, with all of the appropriate options and variables already selected.

In this model, administrators can create separate discovery definitions for each discovery process they may need, with each discrete discovery definition storing the options and variables that are required for a specific discovery process to complete successfully.

For example, administrators can create a discovery definition that simply probes the devices on the local network for a handful of well-known TCP and UDP services, and another discovery definition that probes the devices on a remote network for SNMP management data, with each of these discovery processes subsequently performing different types of synchronization with the configuration database.

Since a discovery definition describes an entire discovery operation from beginning to end, it incorporates all of the options and variables that are needed for the full process to run through completion. More specifically, each discovery definition includes the options and variables that govern the address ranges to be probed, the discovery methods to be used for probing the network addresses, the automation schema definition to be used for processing the results, and the automation action that should be taken upon completion. Some of these options are stored in the discovery definition object itself, while some of them are defined in secondary objects and then referenced by the discovery definition.

The relationship of the major elements of a discovery definition is illustrated in the figure below, and discussed in detail in the following text.

The important point here is that all of these options and variables must be specified before a discovery definition can be created or executed.

Address Ranges - Each discovery process is run against one or more IP address specifications. Address ranges are independent objects that are referenced by the discovery definition, but they are defined as part of the discovery definition, or are defined as part of a discovery method that is also associated with the discovery definition (see next section). At least one address range must be specified, but administrators can define as many address ranges as may be needed. Administrators can also enable and disable address ranges on a per-process basis, which is useful for testing a discovery definition against a small network before it is used in production against multiple large networks.

Discovery Methods - The discovery service is able to use multiple kinds of networking technologies in order to detect the services that are active on each device. Discovery methods are independent objects that are referenced by the discovery definition and are also managed separately. At least one discovery method must be defined for each discovery definition, but multiple discovery methods can be associated with a single discovery definition, and each of the discovery methods can also be enabled or disabled for any given discovery process.

By default, GroundWork Monitor uses a discovery method based on the open source nmap utility to probe well-known TCP ports for common Internet services, and also uses SNMP queries to probe discovered devices for common management data. However, the discovery subsystem can also use nmap to probe for common UDP services, and is also capable of importing WMI discovery data generated elsewhere. Administrators can also create their own discovery methods that use arbitrary scripts and commands if needed.

Automation Schema Definition - Once the process of probing the selected networks with the selected discovery methods has completed, the process of matching the discovered devices with host and service profiles begins. This process is governed by the automation subsystem, which uses its own configuration options that are stored in a separate schema definition object (refer to the Automation section below for more information about schema definitions). The schema definition is used to tell the matching process about the format of the discovery data, as well as describing the string syntax to use when mapping the discovery output to the appropriate host and service profiles. Each discovery definition must have just one schema definition.

Automation Action - Once the profile-matching process completes, the Auto Discovery service can either present the results to the administrator for review, or it can add the discovered nodes to the configuration database automatically, and it can even update the configuration database if so desired. This option is controlled by the automation action setting, which offers a choice between *Interactive*, *Automatic*, and *Auto-Commit* (in the order described above). This option can also be overridden on a per-process basis, thereby allowing the administrator to test a discovery process before having the results automatically committed to GroundWork Monitor's configuration database.

2.0 Managing Discovery Definitions

Discovery definitions are managed with the Discovery console screen. To access the Discovery console screen, select the Auto Discovery menu item from the main menu. The Discovery screen is the default active screen, but you can also explicitly activate this screen by choosing the Discovery menu item in the top menu bar.

2.1 Discovery Console Screen

The default Discovery console screen is shown in the figure below. As can be seen in that figure, GroundWork Monitor provides a default discovery definition called *GroundWork-Discovery-Pro*.

When multiple discovery definitions exist, the currently selected discovery definition (e.g., **GroundWork-Discovery-Pro**) is highlighted with a yellow background. When only one discovery definition exists, it is always selected and therefore always highlighted.
Some of the run-time options and variables that are used for each discovery process are also shown in the main discovery console screen. For example, the available automation **Control Types** (*Interactive, Auto or Auto-Commit*) are shown in the same row as the discovery definition name and description, with the currently defined setting being pre-selected. Meanwhile, all of the known address **Ranges and Filters** are shown in the lower portion of the screen, with the address ranges that are linked to the current discovery definition being preselected.

The automation action and the address ranges can be changed before starting the discovery process, simply by selecting the desired options. By providing access to these options in the main discovery console screen, administrators can modify these settings at run-time without having to directly edit the discovery definition. The default selections for the currently selected discovery definition can be permanently changed by clicking the **Save** button in the upper right corner. Reference for *Auto Discovery* can be access directly using the **Help** button and by clicking "**?**" symbol next to various content. New discovery definitions can be created by clicking the **New** button. This topic is discussed in the *Creating Discovery Definitions* section. Granular modifications to the discovery definition can be made by clicking the **Edit** button. This topic is discussed in the *Editing Discovery Definitions* section. And, to execute a discovery process using the currently selected discovery definition and run-time options, click the **Go>>** button. This topic is discussed in the *Running Discovery Processes* section.

Figure: Discovery Console

2.2 Creating Discovery Definitions

1. To create a new discovery definition, click the **New** button on the discovery console screen. A new screen like the one above will be displayed.
2. Fill in the fields to suit your requirements.
3. Once the fields have been filled in to your satisfaction, click **Create** to continue. At this point the discovery definition editor screen will be displayed, with the new discovery definition values already loaded. If you do not wish to continue creating a new discovery definition, click **Cancel** to return to the main discovery console screen.

New Auto Discovery Definition Fields

- **Auto discovery definition name** - Enter a short name for your new discovery definition.
- **Description** - Enter a description of your new discovery definition.
- **Import/update automation schema** - This drop-down list shows all of the automation schema definitions that have been created. By default, this list will show *GroundWork-Discovery-Pro*. If you need to use a different schema, you can cancel the current operation and return to this dialog after the appropriate schema definition has been created, or you can continue with the current operation and change the discovery definition later. You must choose an automation schema definition to continue creating a discovery definition. Note that discovery processes can only use schema definitions that have a host-import schema type, although all of the schema definitions regardless of type will be shown in the drop-down list. Refer to the Automation section for information about automation schema definitions and schema types.
- **Default control type** - This drop-down list shows the three automation control types that are available. *Interactive* means that the administrator will be required to confirm the results of the discovery process, while *Auto* means that the results will be added to the GroundWork Monitor configuration database (but the configuration will not be automatically committed), while *Auto-Commit* means that the results will be added to the configuration database, the changes will be automatically committed, and GroundWork Monitor will immediately begin monitoring those devices. You must choose an automation action to continue, although you can revise your selection later.
- **Create from template** - Discovery definitions can be saved as *templates* that allow them to be partially reused (see the discussion in the next section for more details). If you have already saved a discovery definition as a template, you can reuse some of its options and variables by selecting it in the drop-down list. By default, this list box will contain a single entry for the *GroundWork-Default-Pro* template which is a template that is derived from the default discovery definition. If you do not want to inherit any options from any other definitions, leave this drop-down list empty.

Figure: Discovery Definition

2.3 Editing Discovery Definitions

To edit the options and variables associated with a discovery definition, select the discovery definition that you want to edit in the discovery console screen. Select the **Edit** button.

1. This will result in the discovery definition editor screen being displayed with the discovery definition values already loaded (this screen will also be loaded automatically after a new discovery definition is created, as described in the preceding section). The figure shows the discovery definition editor screen, using the values from the default **GroundWork-Discovery-Pro** discovery definition.
2. Once you have finished making changes to the discovery definition, select one of the following options click the **Save** button to make the changes permanent;
   or
   If you would like to create a template of the discovery definition that can be applied to future definitions, click the **Save As Template** button. Templates are stored as XML text files in the `/usr/local/groundwork/core/monarch/automation/templates` directory on the GroundWork Monitor server. Template files can be copied into the same directory on another GroundWork Monitor server and will be immediately usable on that system;
   or
   If you wish to initiate a discovery process using the current settings but without making the changes permanent, click the **Go >>** button. If you want to cancel your changes and return to the main discovery console screen, click the **Close** button.

Figure: Editing Definitions

Discovery Definition Fields

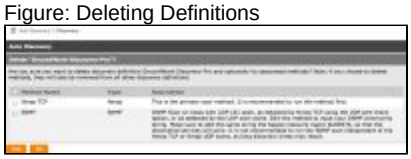
- **Description** - Enter a brief description for this discovery definition.
- **Import/update automation schema** - This drop-down list shows all of the automation schema definitions that have been created. Note that discovery processes can only use schema definitions that have a host-import schema type, although all of the schema definitions regardless of type will be shown in the drop-down list. Refer to the Automation section below for information about automation schema definitions and schema types.
- **Default control type** - This drop-down list shows the three automation control types that are available. *Interactive* means that the administrator will be required to confirm the results of the discovery process, while *Auto* means that the results will be added to the GroundWork Monitor configuration database automatically (but the configuration will not be committed), while *Auto-Commit* means that the results will be added and the changes will be committed.
- **Traceroute Option** - This option allows the discovery service to use the traceroute command to automatically determine the parent device of a discovered device. This option is only useful if remote networks are being probed and if the host entries in Nagios have parent-child relationships that reflect network topology. Note that the traceroute command requires ICMP to be enabled on the discovered devices and the intermediary network devices. Also note that the default hop-count and timeout values are tuned for moderately-sized networks, and you may need to override the default values if you intend to probe for devices across a very large or slow wide-area network.
- **Methods** - This portion of the editor screen shows the discovery methods that will be used to probe discovered devices for their available network services. All of the discovery methods that have been defined are displayed here (including methods that have been created for other discovery definitions), while the discovery methods that are associated with the current discovery definition will be actively selected. On a new installation, this area will only show the *Nmap TCP* and *SNMP* discovery methods that are provided with the default installation, although additional discovery methods may be created as needed, and the existing discovery methods may also be modified. New discovery definitions will not have any associated discovery methods unless the administrator used a discovery template during the creation process that had one or more methods already selected. Refer to the Managing Discovery Methods section below for information on creating and managing discovery methods.
- **Ranges and Filters** - This portion of the editor screen shows the address ranges that will be probed. All of the address ranges that have been defined are displayed here, including those that have been created for other discovery definitions. On a new installation, this area will only show a *local subnet* address range that was created during installation, although additional address ranges may be created as needed, and the existing address ranges may also be modified. New discovery definitions will not have any associated address ranges unless the administrator used a discovery template during the creation process that had one or more address ranges already selected. Refer to the *Managing Address Ranges* section below for information on creating and modifying the address ranges.

2.4 Deleting Discovery Definitions

If you want to delete a discovery definition, you must do so from inside the discovery definition editor screen. Select the discovery definition that you want to delete from the main discovery console screen, and click the **Edit** button. Once the discovery definition editor screen has loaded, click the **Delete** button in the upper right corner. You will then be presented with a confirmation box similar to the following, which will also allow you to delete any discovery methods that may be associated with the discovery definition:

If you want to delete any of the discovery methods associated with the discovery definition, activate the appropriate check box. Once you are satisfied with your choices, click the *Yes* button to delete the discovery definition and the selected discovery methods, or click the *No* button to cancel and return to the discovery definition editor screen.

Figure: Deleting Definitions



3.0 Managing Address Ranges

Address ranges are used to specify the IP addresses that should be probed during the discovery process. Address ranges can be specified in the main discovery console screen, the discovery definition editor screen, and the discovery method editor screen. In all these cases, a portion of the screen will have a **Ranges and Filters** section similar to the figure below.

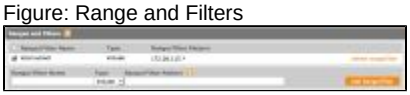
During installation, GroundWork Monitor examines the server's local IP address and subnet mask and attempts to create a *local subnet* address range that reflects the local network's properties. This is shown as the *local subnet* entry with the Range/Filter value of *127.0.0.** in the example below, although your own entry should reflect the IP addressing in use on your local network.

3.1 Create a New Address Range

You can create as many address ranges as are needed, and multiple address ranges can be assigned to any discovery process. At least one address range must be active in order for a discovery process to execute.

1. To create a new address range, fill in the fields with appropriate values and click the **Add Range/Filter** button.
2. At this point the newly created address range will appear in the **Ranges and Filters** section of the current screen. Remember that address ranges must be selected before they will be incorporated into a discovery process.
3. To delete an address range, click the **delete range/filter** hyperlink next to it. There is no edit function for address ranges. If you wish to change an address range, it must be deleted and recreated.

Figure: Range and Filters



Ranges and Filters

- **Name** - Enter a short name for the address range.
- **Type** - There are two types of address ranges in Auto Discovery: an *include* range that specifies a range of addresses that should be probed, and an *exclude* range that specifies a subset of addresses that should not be probed. IP addresses that are not part of an *include* range are never probed, so you do not need to *exclude* any addresses unless they fall within the scope of an *include* range.
- **Range/Filter** - Address ranges can use any of the following different formats:
    - Unqualified hostname (e.g., mybox)
    - Qualified hostname (e.g., mybox.mydomain.com)
    - Single address (e.g., 192.168.0.1)
    - Address range (e.g., 192.168.0.2 - 192.168.0.10)
    - Abbreviated address range (e.g., 192.168.42-50)
    - CIDR block (e.g., 192.168.144.0/20)
    - Comma-separated list of any of the above (e.g., 192.168.0.1, 192.168.0.3)
      In Class C specifications, the network (.0) and broadcast (.255) addresses will be automatically ignored. Equivalent addresses will be ignored in CIDR blocks with a subnet prefix smaller than /31.
      If you are using a discovery script to generate a list of hosts, you need to put a dummy value into this field in order to make the discovery proceed.

4.0 Managing Discovery Methods

Discovery methods are used to specify the network services that should be probed for on each IP address. During this process, the discovery service iterates through the list of IP addresses that have been associated with the current discovery definition, executes one or more tests against the current address, and then generates one or more lines of output data that catalogs the services that have been discovered. The output data is then subsequently read and analyzed by the automation subsystem, which maps the host and service data to the appropriate Nagios profiles.

> ⚠ Discovery methods are executed sequentially, and in some cases will only query devices that were discovered by a previous discovery method. For example, in the figure above the *Nmap TCP* discovery method will be executed before the *SNMP* discovery method (this relationship is important to the SNMP discovery method, as will be explained later). You cannot currently reorder discovery methods, so they must be created in the order desired.

Discovery methods can only be associated with a discovery process in the discovery definition editor screen, which has a *Methods* section similar to the following. GroundWork Monitor provides two discovery methods by default, both of which are shown in the figure:

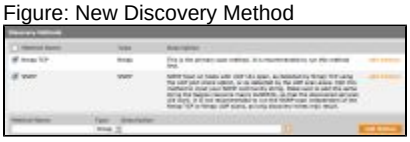
- The *Nmap TCP* discovery method is used to probe devices for common TCP/IP network services such as Web and email services.
- While the *SNMP* discovery method is used to probe for SNMP management information.

You can modify either of the bundled discovery methods, or you can create your own discovery methods if needed. Multiple discovery method can be assigned to each discovery definition. At least one discovery method must be active in order for a discovery process to execute.

4.1 Create a New Discovery Method

1. Fill in the fields with appropriate values and click the **Add Method** button.
2. At this point a discovery method editor screen will be displayed that will allow you to set all of the options and variables associated with the discovery method (the layout of the discovery method editor screen is different for each discovery type).
3. If you want to modify an existing discovery method, click the **Edit Method** hyperlink next to the discovery method. This will cause the appropriate discovery method editor screen to be loaded with the contents of the selected discovery method.

Figure: New Discovery Method



Discovery Method Fields

- **Name** - Enter a short name for the discovery method.
- **Type** - There are four types of discovery methods available, which are described in more detail below. However, it is important to understand that this field is only used to select the software that actually probes the devices, and additional configuration will be required in a separate screen before the discovery method is fully functional.
    - *Nmap* - This type is used when you want to have the discovery process probe for TCP or UDP network services on the discovered devices. This is the primary discovery method in GroundWork Monitor, and should be used as the first discovery

method in most cases. Note that the nmap utility is sometimes classified as an intrusion tool by some security tools and/or personnel, and you may need to coordinate with other groups before using it.
    - *SNMP* - The SNMP type is used when you want to have the discovery process probe for SNMP management information on the discovered devices.
    - *Script* - If you want to create your own probes with your own script commands, use the *Script* type.
    - *WMI* - This type is used when you want to import Windows-specific host data into GroundWork Monitor. The WMI data is generated by a separate Windows-based program.
- **Description** - Enter a brief description of the discovery method.

4.2 Nmap Discovery Method

The *Nmap* discovery method uses TCP and UDP connection requests to probe for network services on each discovered device. Whenever a probe succeeds, the discovery method returns a service *match value* that indicates the service(s) that were discovered, along with host platform identification information (if available). Once all of the probes have completed, the match values will then be subsequently correlated with the appropriate host and service profile(s) for that device by the automation processor.

Multiple nmap service probes can be defined, but they must all be of the same protocol type, and at least one service must be defined.

The figure below shows the discovery method editor for the *Nmap* type, using the default *Nmap TCP* discovery method. Also, the options and fields in the nmap discovery method editor screen are shown in the table below (note that different fields will be used for different types of nmap probes).

Once the options and fields have been completed to your satisfaction, click the *Save* button in the upper right to continue. At this point you will be returned to the discovery definition editor screen that you came from. You can also rename or delete the discovery method by clicking on the appropriate button.

Figure: Nmap Discovery Method

Nmap Discovery Method Editor Options and Fields

- **Scan Type** - There are three nmap scan types available, which are as follows:
    - *TCP SYN Scan* - Network services that use TCP require a full synchronization *handshake* before data can be exchanged. With the TCP SYN scan method, nmap generates raw connection request packets for the target service(s), but does not fully complete the handshake procedure. Instead, it only waits to see if the connection request is acknowledged or refused before moving to the next TCP port number in the list. Although this method is very fast, it requires custom packets and as such it is only available when the discovery process is run under an account that has root user privileges (GroundWork Monitor uses the necessary privileges by default).
    - *TCP Connect Scan* - The TCP connect scan method uses standard user-level operating system calls to perform a complete handshake operation with the target TCP service. This method takes longer to successfully determine the results of the connection request, but it works with any user account.
    - *UDP Scan* - The UDP scan method is for detecting UDP based network services. UDP is a stateless transport protocol and does not have a handshake mechanism, so nmap simply looks for data of some kind and assumes that any response activity is from the expected application.
- **UDP SNMP Check** - This option determines whether or not a discovered device is also probed with a UDP query to see if SNMP is enabled on the target. If this option is enabled, and if the SNMP discovery method has also been enabled, then only those devices that respond to the UDP query will be probed by the subsequent SNMP discovery method probes. Note that this checkbox is only available when one of the TCP scans have been selected (UDP scans can explicitly probe for SNMP without requiring a separate option).
- **SNMP Match Strings** - During the TCP scanning process, nmap attempts to identify the host platform and operating system in use on the target device by examining certain characteristics of the TCP protocol handshake, with this information then being recorded in the result data. When this process is successful, the resulting host identifier string can be used as an additional input filter to further limit the number of follow-up SNMP queries. In order to use this option, you must enter a filter string in this field that will match some or all of the device identification strings that you want to use as a limiting filter. Typically, this will be a string such as ”r;cisco” or ”r;hewlett-packard” or some other substring of an nmap host identifier. Multiple strings can be entered but must be separated with a comma. Matches are case-insensitive. Note that this field is only enabled when one of the TCP scans are used (nmap does not have the ability to detect the host operating system with UDP probes, since UDP does not have a handshake).
- **Scan Timeout** - This option determines the delay between the connection requests that are generated by nmap. Shorter delay periods

result in faster scans, but will also timeout quickly when the local GroundWork Monitor host and the remote systems are separated by a long or slow network path. Aggressive delay periods are also known to set off intrusion alarms. The options for this field are *paranoid*, *polite*, *normal*, *aggressive*, and *insane*, which range from very slow to very fast in the order presented. The default value is *insane* and will yield the fastest results, but may result in missed probes or security problems on some networks.

- **Ports** - This section of the nmap discovery method editor screen is used to enumerate the port numbers that should be probed and the *match value* that should be returned when a probe is determined to have succeeded.
  To create a new assignment, enter a port number and a match value to be returned upon success, and click the *Add Port* button. To delete an existing assignment, click the *delete port* hyperlink next to an entry.
- **Match Values and Actions** - The match value string will be stored in the resulting log file, where it will then be used to correlate service profiles with the host entry. Note that match values are keyed to existing services, service profiles, host profiles and **discover** values. You are not restricted to these keyed values, though you will be presented with an ordered list of the services, service profiles, host profiles, and discover match values available in your configuration database, or any profiles loaded in the `/usr/local/groundwork/core/profiles` directory on the GroundWork server. The effect of selecting one of the keyed service, service profile, or host profile values is determined by the automation schema used. See the section on Automation for details of editing the schema. The default schema will process these as follows:

  
  Newly created match values will not be useful until the appropriate automation schema definition has also been updated (refer to the Automation Subsystem section for more information on this process).

  - **service-<service name>** - Assign the named service to the discovered host.
  - **service-profile-<service profile name>** - Assign the named service profile to the discovered host.
  - **host-profile-<host profile name>** - Assign the named host profile to the discovered host.
    The match values will have effect only on subsequent methods. The effect of the different match values is as follows:
  - **discover-snmp** - Subsequent SNMP methods will be restricted to hosts that match this port signature (as well as those that match the String and UDP scan options described above).
  - **discover-script** - Subsequent script methods will be restricted to hosts that match this port signature.
  - **discover-wmi** - No effect. This is reserved for future functions.
- **Ranges and Filters** - This section of the discovery method editor screen allows you to link the discovery method to a specific address range. If a discovery definition will use multiple discovery methods for multiple address ranges, and if the current discovery method is only useful for some of the address ranges, then you may wish to enable a specific address range for this discovery method. However, in those cases where the discovery definition uses all of the defined discovery methods with all of the active address ranges, no address ranges should be explicitly enabled in the discovery method.

4.3 SNMP Discovery Method

The **SNMP** discovery method uses SNMP queries to see if a device supports the SNMP protocol. If so, additional probes are used to enumerate the network interfaces on that device, with each interface subsequently being assigned an appropriate service profile. If the device does not respond to the SNMP queries, it is ignored.

In addition, devices that respond to SNMP are optionally queried for specific metrics. These metrics are specified outside the user interface in a configuration file, located at `/usr/local/groundwork/core/monarch/automation/conf/snmp_scan_input.cfg`.

The format of this file is OID=matchstring where OID is a numeric SNMP object identifier, and matchstring is an arbitrary string used to map services or service profiles to hosts that are capable of responding to SNMP queries for the specified OID.

This is an advanced option. Configuration requires knowledge of the SNMP MIB on the device to be probed. GroundWork Monitor has several examples supplied in the default file. The necessary schema values required to process these samples are included in the default automation schema for the default discovery definition in GroundWork Monitor.

Note that an SNMP discovery method that is associated with a discovery definition will be used with all of the address ranges associated with the current discovery process. SNMP queries use UDP, which does not provide a handshake mechanism, and therefore requires a timeout interval to detect failure (the default value is 15 seconds). This means that SNMP probes can be very time consuming, especially on large networks.

For this reason, Auto Discovery allows SNMP discovery methods to be linked to TCP scans, which are much faster. In this arrangement, only those devices that have been successfully discovered with TCP probes are also probed with follow-on SNMP queries (the Nmap TCP method contains a quick UDP scan option for this purpose). Furthermore, administrators can also limit queries to specific types of host platforms, thereby forgoing the need to enumerate the network interfaces on all devices. For more information about this process, refer to the Nmap Discovery Methods section above.

The figure shows the discovery method editor for the *SNMP* type, with no preloaded values. Once the options and fields have been completed to your satisfaction, click the *Save* button in the upper right to continue. At this point you will be returned to the discovery definition editor screen that you came from. You can also rename or delete the discovery method by clicking on the appropriate button.


SNMP community strings are not passed from Auto Discovery to Auto Configure since community strings are often security sensitive. It is recommended that custom Auto Discovery mechanisms that need to distinguish between available community strings match against OIDs that are present in one community string but not the other.

Figure: SNMP Discovery Method

SNMP Discovery Method Editor Options and Fields

- **SNMP Version** - The SNMP discovery method supports SNMP versions 1, 2c, and 3. Version 1 of SNMP is the simplest and has the widest support, but also lacks some of the efficiency in version 2c. SNMP version 3 offers the highest security, but is not as widely supported as other versions. By default, the SNMP discovery method uses SNMP version 2c which is the most efficient and has wide support. Version 2c should be used unless you know that your network devices are configured to use version 3.
- **SNMP v3** - The SNMP v3 fields are disabled unless SNMP version 3 is selected. If you use version 3, you will need to provide the authentication information in the SNMP v3 fields.
- **Community Strings** - This field is disabled unless version 1 or 2 is selected. If you use version 1 or 2, provide one value in the Community Strings field to be used with the lookup queries.
- **Ranges and Filters** - This section of the discovery method editor screen allows you to link this specific discovery method to a specific address range. If a discovery definition will use multiple discovery methods for multiple address ranges, and if the current discovery method is only useful for some of the address ranges, then you may wish to enable a specific address range for this discovery method. However, in those cases where the discovery definition uses all of the defined discovery methods with all of the active address ranges, no address ranges should be explicitly enabled in the discovery method (they will be included by the discovery definition at run-time instead).

4.4 WMI Discovery Method

The *WMI* discovery method uses independent scans of the Windows Management Interface to probe Windows devices for their disk, memory, and CPU resources. These scans require a GroundWork Windows Child Server. The figure shows the discovery method editor for the *WMI* type, with no preloaded values. The WMI discovery method editor screen only provides one option, which is the WMI Type that should be probed for. There are two WMI scan types available, which are as follows:

WMI Discovery Method Editor Options and Fields

- **Server** - This scan type loads all discovery output available from GroundWork Windows Child Servers. The Windows Child Server must be independently configured to upload these results to the GroundWork Server.
- **Proxy** - This option remains unimplemented. Selecting it will have no effect.

> ⚠ There is no address range management area in the WMI discovery method editor screen, since there is no actual scanning.

Once the options and fields have been completed to your satisfaction, click the *Save* button in the upper right to continue. At this point you will be returned to the discovery definition editor screen that you came from. You can also rename or delete the discovery method by clicking on the appropriate button.

Figure: WMI Discovery Method

4.5 Script Discovery Method

The **Script** discovery method is designed to allow administrators to create their own discovery probes. Although the Auto Discovery subsystem provides discovery methods that can locate most of the common network services, sometimes an administrator will want to create their own discovery methods. Towards that end, the Script method can be used to look for almost any kind of device characteristic imaginable, with the only requirements being that the Groundwork host is able to execute a command and return the necessary output, and that Nagios is able to use a service definition that appropriately describes the desired resource.

For example, an administrator may want to catalog the hosts that are currently reachable on the network, using a tool such as ping to discover and monitor those hosts, instead of using TCP and SNMP probes to monitor all of the services running on all of the hosts. In this scenario, an

administrator could use a simple script that called the ping utility and then generated the necessary output for each IP address, with the result data subsequently being used to associate the correct Nagios host and service profiles with the discovered devices. Once this script was associated with a custom discovery method, the administrator could perform the ping discovery process whenever needed.

By the same token, these same principles can also be used to measure other types of network services, such as discovering devices on AppleTalk or DECnet networks, or discovering local applications that are running on remote hosts, simply by creating the necessary script and mapping the output to the appropriate Nagios profiles.

In order to clarify these points, the script below represents a fully-functional discovery script that uses the Nagios `check_icmp` utility to ping a target device, looks up hostname data associated with the target, and then generates an appropriately formatted response line based on the discovered data.

```
#!/usr/local/groundwork/perl/bin/perl
#
# ping_discovery.pl
# v1.0
#
# Copyright 2008 GroundWork Open Source, Inc.
use strict;
use warnings;
use Net::hostent;
use Socket;
# first make sure we got a valid IP address as the first parameter
if (! defined $ARGV[0]) {
die "Fatal error: input does not include a valid IP address \n";
}
if ($ARGV[0] !~ /^(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})$/) {
die "Fatal error: input does not include a valid IP address \n";
}
# predefine all variables
my $hostaddr = $ARGV[0];
my $ping = "";
my $host = "";
my $hostname = "";
my $hostalias = "";
my $hostprofile = "";
my $svcprofile = "";
my $service = "";
my $output = "";
# ping the specified IP address and store the results
$ping = qx(/usr/local/groundwork/nagios/libexec/check_icmp -H $hostaddr) or die;
# if we get an OK or WARNING response, start looking for hostname information
if (($ping =~ /^(OK.+)\|/) || ($ping =~ /^(WARNING.+)\|/)) {
# clean the test results
$ping = $1;
chomp ($ping);
# see if any hostname information is defined for the host
if ($host = gethost($hostaddr)) {
# if the hostname is an FQDN, use the short version
if ($host->name =~ /^(\S+?)\./) {
$hostname = $1;
}
else {
$hostname = $host->name;
}
# if an alias is defined and is different from the short
# hostname, use it
if ((@{$host->aliases}) && (@{$host->aliases}[0] ne $hostname)) {
$hostalias = @{$host->aliases}[0];
}
# otherwise, if the original hostname is an FQDN, use that
# for the alias
elsif ($host->name =~ /^(\S+?)\./) {
$hostalias = $host->name;
}
# if all else fails, reuse the hostname for the alias
else {
$hostalias = $hostname;
}
```

```
}
# host does not have a known hostname, so reuse the IP address
else {
$hostname = $hostaddr;
$hostalias = $hostaddr;
}
# set the Nagios profile data
$hostprofile = "host-profile-service-ping";
$svcprofile = "service-ping";
$service = "icmp_ping_alive";
# generate the appropriate output for the automation schema definition
# name;;alias;;address;;description;;parent;;profile;;service profile;;service
$output = $hostname . ";;" .
$hostalias . ";;" .
$hostaddr . ";;" .
$ping . ";;" .
";;" .
$hostprofile . ";;" .
$svcprofile . ";;" .
$service ;
}
# host did not respond to ping, so return empty response data
# name;;alias;;address;;description;;parent;;profile;;service profile;;service
else {
$output = ";;" .
";;" .
";;" .
";;" .
";;" .
";;" .
";;";
}
```

```
# return the output
print $output . "\n";
exit;
```

As can be seen, the example script reads the input for an IP address that has been passed to it by Groundwork during the discovery process, and then calls the Nagios `check_icmp` command to ping the target address. If the target device responds to any of the probes (as determined by an OK or WARNING command response), then a series of hostname lookups are performed in order to determine the device's name and alias. Finally, the host and service profile attributes are filled in with canned values, and the response data is returned. Meanwhile, devices that did not respond to any of the probes successfully are simply ignored, and a row of empty values are returned.

The example script is intended to be executed separately for each target IP address, with each individual address being passed to the script by GroundWork Monitor according to the address ranges that were registered with the discovery definition.

In order for GroundWork to successfully use the script, the script output must correlate with the field layout defined in the automation schema definition that is associated with the current discovery definition. For example, the script above is intended to be used with the default GroundWork schema definition, which has a predefined layout and matching rules. If the default schema definition is used, the output file will contain the same fields as described in the Automation Data Files section below.

In particular, it is crucial that every explicit host record contain a host name and IP address field in order to indicate the specific host that the record applies to, or else the automation processor will infer that the current record is an update to the previous host record and will append or overwrite some of the field data. Similarly, every host that is probed must generate a line of response data in order for the discovery processor to know that the script has performed its task. This may either be a row of completely blank fields (as in the sample script above), or a row that contains the host name and address which indicates that the record data applies to an explicit host. Refer to the discussion in The Automation Process section below for more information about the file layout rules and how the contents of the file are analyzed and used.

The script output is captured by GroundWork and stored in a file in the `/usr/local/groundwork/core/monarch/automation/data` directory, with the file name reflecting the name of the discovery definition. The data in the results file will only be processed by the automation subsystem after the discovery process has completed. Once a script has been created, it must be registered as a discovery method before it can be used. The figure shows the discovery method editor for the *Script* type, with no preloaded values.

Once the options and fields have been completed to your satisfaction, click the *Save* button in the upper right to continue. At this point you will be returned to the discovery definition editor screen that you came from. You can also rename or delete the discovery method by clicking on the appropriate button.

Figure: Script Discovery Method

Script Method Editor Options and Fields

- **Script type** - The only currently supported type is the *Custom* type.
- **Run Mode** - The *Run Mode* type determines whether the script is executed for every IP address, or if it is only executed one time during the discovery process. If the *Batch Mode* checkbox is disabled then the script will be executed for every IP address, but if the checkbox is enabled then the script will only be executed one time.
- **Command Line** - You must provide the full path to the command that will be used for the discovery method. The script file must be stored in the `/usr/local/groundwork/core/monarch/automation/scripts` directory, and the `nagios` user account must have sufficient permissions to execute the script.

  You can specify parameters and arguments to the script in this field, either as hard-coded arguments or environment variables that are usually available to the nagios user account. If the script is being executed for each host address, the variables can also refer to Nagios macro definitions such as $HOST$ or $user21$, with these macros being expanded by GroundWork Monitor before they are passed to the command line. However, if the script is being executed in batch mode, the macros are not expanded, and will not be available to the script.

- **Ranges and Filters** - This section of the discovery method editor screen allows you to link this specific discovery method to a specific address range. If a discovery definition will use multiple discovery methods for multiple address ranges, and if the current discovery method is only useful for some of the address ranges, then you may wish to enable a specific address range for this discovery method. However, in those cases where the discovery definition uses all of the defined discovery methods with all of the active address ranges, no address ranges should be explicitly enabled in the discovery method (they will be included by the discovery definition at run-time instead).

4.6 Deleting Discovery Methods

If you want to delete a discovery method, you must do so from inside the discovery method editor screen.

1. Go to the discovery definition editor screen, and click the **edit method** hyperlink next to the discovery method that you want to delete.
2. Once the discovery method editor screen has loaded, click the **Delete** button in the upper right corner. You will then be presented with a confirmation box similar to the figure below.
3. Click **Yes** to delete the discovery method, or click the **No** button to cancel and return to the discovery method editor screen.

Figure: Deleting Discovery Methods

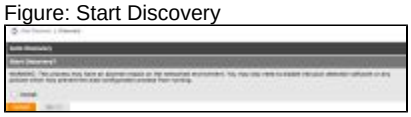
5.0 Running Discovery Processes

A discovery definition can be executed from either the main discovery console screen or the discovery definition editor screen simply by clicking the *Go>>* button in the upper right corner. Note that you can change the automation action or enable and disable address range(s) at run-time, and those changes will be incorporated into the selected discovery process when it starts, but all other options can only be changed in the discovery definition editor.

When the discovery process starts, an output file is created in `/usr/local/groundwork/core/monarch/automation/data` with a filename that is based on the name of the discovery definition being used. If another instance of the selected discovery process has already been executed but its output data has not yet been fully processed, this file will already exist, and you will be presented with a screen where you may choose to discard the previous output and begin a new discovery job, or you may choose to forgo the discovery process and skip directly to the automation process, or you may cancel the request and return to the main discovery console screen.

If you are still proceeding with the discovery execution, you will next be presented with a screen similar to the following, which warns that the discovery operation may trigger intrusion-detection software or set off other kinds of security alarms. Note that you may need to coordinate with network security personnel before conducting these operations in order to avoid undesirable problems that may result from these false alarms. You must acknowledge the warning by activating the *Accept* toggle and clicking the *Go>>* button before the discovery operation will proceed.

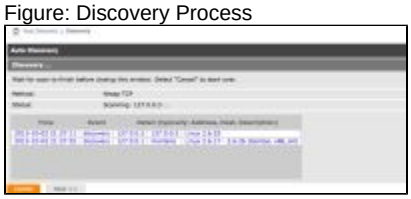
Figure: Start Discovery



Once the discovery process begins, you will then be presented with a status screen similar to the following. This screen will be updated in real-time as each host is probed. At this point, you will be unable to change any details unless you cancel the current operation and start over. The amount of time required to complete the discovery process will depend on numerous factors, such as the size of the address range(s) and the type of discovery method(s) in use, the network latency between the GroundWork Monitor server and the remote devices, and a variety of other factors. As a general rule of thumb, it takes approximately 10 minutes to scan a small network with the default discovery definition and all of its associated options.

After all of the IP addresses have been probed, the *Next>>* button will become accessible, and you may then proceed with the automation process that will match the discovered devices to their appropriate host and service profiles and perform any necessary database updates. This subject is discussed in the Automation Subsystem section.

Figure: Discovery Process



Part II: The Automation Subsystem

1.0 About The Automation Subsystem

In practice, the discovery process is relatively straightforward, and usually only requires an administrator to answer a few confirmation prompts. However, there are a large number of options and variables that must be defined before this process can be started.

Apart from the discovery subsystem (described in the preceding section), the Auto Discovery service also provides an *Automation* subsystem that can be used to import structured data into GroundWork Monitor's configuration database. Using this subsystem, administrators can have the Auto Discovery service read input data from a simple text file and synchronize the configuration database with that data.

In fact, the automation subsystem is used for just this purpose by the discovery subsystem whenever the results of a discovery process need to be imported into the configuration database. However, the automation subsystem is not limited to being used in this manner, and can be used to

synchronize the configuration database with any data source that is able to generate a suitable input file.

GroundWork Monitor uses schema definitions to store the configuration options for a particular automation process, similar to the way in which it uses discovery definitions to store options and variables for a discovery process. In this model, administrators can create separate automation definitions for each of the data-import processes they may need, with each discrete schema definition storing the options that are required for a specific automation process to complete successfully.

For example, administrators can create a schema definition that tells the automation subsystem to read a discovery log for host names and service definitions, and then update the existing configuration database entries with the discovered data. Meanwhile, another schema definition can be created that tells the automation subsystem to recreate the entire configuration database from the input data, which may be useful when the configuration information is being fed from an outside management system. Once these schema definitions have been created, administrators can simply select the most appropriate schema definition for the task at hand and immediately begin the associated automation process, with all of the appropriate options already selected.

As with discovery definitions, a schema definition describes an entire import operation from beginning to end, and incorporates all of the options that are needed for the full process to run through completion. More specifically, each schema definition includes the options that govern the type of synchronization to be performed, the input source file, a description of the input file's layout, and the processing rules that should be applied to the input data. These elements are discussed in detail in the following sections.

2.0 The Automation Process

In simple terms, the automation subsystem reads in lines of text from an input file, and then breaks each line into fields of data. As each field is extracted from the input file, one or more matching rules dictate what will happen with that piece of data next (it may be mapped to a field in the GroundWork Monitor configuration database, or it may be discarded, and so forth). Once all of the input data has been parsed and analyzed, the automation subsystem can present the final results to the operator for review, and then updates the configuration database with the final results.

In order for this process to work properly, GroundWork Monitor requires the input text to be line-oriented, with each line of input data representing a single row. However, GroundWork Monitor does not require the input data to have fixed-width columns, but instead assumes that the data uses a fixed number of variable-length fields. In this model, each field of data in the input file is expected to be separated from its neighbors by a user-defined sequence of characters, which can either be a single character such as a simple comma (for importing CSV files), or can be multiple characters, and it can even use regular expressions if needed.

Administrators must define the fields that are present in each row of data, along with the processing rules that will be applied to each field as the data is extracted. As the automation subsystem reads each field, the data is mapped to a temporary grid of rows and columns that is conceptually similar to a spreadsheet. In this model, each row of data contains a single record, while each column identifies a specific piece of data, with the intersection of a row and column (a cell) containing each individual piece of data from each record.

Meanwhile, each logical column has one or more processing rules attached to it that tell the automation subsystem what should happen to when a specific field is extracted from the input file. These rules may manipulate the data, discard the data, or map the data to a field in the GroundWork Monitor database, among other things. Each column can have multiple processing rules, and each rule will be executed in sequence until all of the rules for that column have been executed.

After all of the input data has been extracted and processed, the final results can be displayed to the operator for confirmation, and inserted into the GroundWork Monitor configuration database.

As an example of how this might work, consider the following sample input:

```
cisco-router, 192.168.10.1, snmp
windows-server, 192.168.10.21, wmi
linux-server, 192.168.10.34, ping
```

In the sample above, the data consists of three rows of text, with each row containing fields for hostname, IP address, and host profile, each of which are separated from their neighbors by a single comma character. In order for this data to be useful to the automation subsystem, a schema definition would need to be created which specified the location of the input file, the field separator in use (a single comma), the logical columns in the source data (name, address, and profile), and the processing rules for each of the columns.

Once an appropriate schema definition was created, the automation subsystem would then use those settings to begin an automation process. In that scenario, the automation subsystem would examine each row of data in the specified input file for the separator character, extract the relevant data from each field, and apply the matching rules that had been defined for each column, with this processing repeating for each individual row of data. Once the entire input file had been fully read, the automation subsystem would then present the results to the administrator for review, before finally updating the configuration database with the extracted data.

Although the simple example above can be made to work with some limited applications, the default schema definition provided with GroundWork Monitor is slightly more complicated, and should be used for processes that need to synchronize host and service profile data with the configuration database. Specifically, the default schema definition uses the following column layout, with each field separated from its neighbor by a pair of semi-colon characters:

```
name;;alias;;address;;description;;parent;;profile;;service profile;;service
```

A host record must include a hostname and IP address field in order for the record to be associated with a specific device. If an input record does not contain both of those pieces of data then the record will be interpreted as an update to the previous record, and the data that is present in the current record will append or overwrite the data collected from the previous record. This model is intentional, and allows multiple discovery methods to add supplemental profile data or fill in host data with additional details as the data is discovered.

For illustration purposes, `/usr/local/groundwork/core/monarch/automation/data/sample-multiline.txt` contains sample input data which demonstrates these concepts. That file is typical of input data that is generated by discovery methods which use the default schema definition.

2.1 Synchronization Methods

A critical aspect of the automation process is the manner in which the input data and the configuration database are synchronized together. In the usual scenario, an administrator will likely want to simply add new records or update existing records, while leaving any other existing configuration entries unmodified. In some cases, however, it may be desirable or necessary to destructively recreate the configuration database entirely, especially if a separate management tool is the primary management point for network resources and assets.

The synchronization behavior is governed by the schema type option. One schema type must be set for every schema definition. This setting affects other configuration options, so the schema type must be set whenever a new schema definition is first created, and the schema type associated with a schema definition cannot be changed after it has been created.

Schema Types

- **host-import** - The host-import schema type can add new configuration entries to the configuration system, and can also update existing host configuration entries, but will not modify any other existing configuration entries.
  The host-import schema type is the only schema type that can be used for discovery processes, and should generally be used for most automation tasks unless another schema type is known to be more appropriate.
- **host-profile-sync** - The host-profile-sync schema type will destructively synchronize the configuration database with the contents of the import file, effectively causing the entire configuration database to be recreated every time a schema definition is executed. This schema type is intended to be used when a separate management tool (such as Cacti) is the primary source of configuration information. Given the destructive nature of this schema type, it should be used very prudently
- **other-sync** - The other-sync schema type will only modify existing configuration entries, and will not add new entries, nor will it destructively synchronize the configuration database. It is intended to be used for making bulk modifications to configuration entries. For example, it can be used to change the contact groups for specific host entries, or to change the contact groups themselves.

2.2 Matching Rules

When the input file is read by the automation process, each input record is analyzed and broken into its discrete data fields, which are then extracted and assigned to their corresponding columns in an intermediary structure. As the contents of each field are assigned to their column, one or more matching rules are applied to data, with these rules dictating what is to become of that data.

In the simplest scenario, the data in a column will be mapped to a field in the GroundWork Monitor configuration database, where it will eventually be used to create or update an entry in the configuration database. However, this is a very simplistic scenario, and there are several other possible courses of action. For example, the default schema definitions include matching rules that discard comments and duplicate hosts, link device types to host groups based on operating system identifiers, and more, with all of this work being performed by different matching rules.

Technically, matching rules are conditional processing rules that are similar to traditional *if-then* rules. The conditional processing is accomplished through the use of string-comparison functions that preface a defined action. For example, a matching rule can be written that says *if the device identifier is 'linux' then associate the device with the 'Linux Servers' host group* while another matching rule can be written that says *if the first character of data is '#' then discard the entire record*.

Once a conditional processing filter has been met, the action part of the matching rule is executed. This is where the real meat of the automation system comes into play. The most common action is a simple statement that instructs GroundWork Monitor to use the field data directly, which is useful for simple tasks such as assigning a host name from the field value. However, more complex actions are also provided, some of which include additional conditional processing filters of their own. For example, some rules will test to see if a configuration object already exists, with different courses of action being taken depending on the outcome of that secondary evaluation.

Each matching rule contains just one string-comparison function, and just one action, although multiple matching rules can be defined for each unique column. In this scenario, each of the different matching will be executed in sequence, until all of the rules have been processed.

More specifically, each matching rule consists of at least four configuration options, although some rules require additional configuration data and therefore have more options. However, the matching filter and the rule directive are the most important two.

2.3 Match Filters

The match attribute stores the conditional processing filter for the current matching rule (IE, it stores the *if* part of the if-then statement). The remainder of the matching rule will only be processed if this comparison returns a *true* result. If the field data does not match the filter, the data will not be processed by the current matching rule, although another matching rule further ahead may be executed if the data matches that rule's filter. The comparison types that are available for use are as follows:

Comparison Types

- **use-value-as-is** - No comparison is performed, and the matching rule is always executed, even if the field is empty. This is essentially the same as always returning a *true* result for the conditional processing.
- **is-null** - The matching rule will only execute if there is no data. This could happen because the original input provided an empty field, or because an earlier matching rule for the column deleted the contents.
- **exact** - The matching rule will only execute if the field exactly matches the specified string. Matching rules are not case sensitive.
- **begins-with** - The matching rule will only execute if the field begins with the specified string.
- **ends-with** - The matching rule will only execute if the field ends with the specified string.
- **contains** - The matching rule will only execute if the specified string is found somewhere within the field.
- **use-perl-reg-exp** - Use this matching type if you want to specify your own Perl regular expression for string comparison. This may be needed if you want to force case-sensitive matching, or if the input data is particularly noisy. Note that you can use a single pair of parenthesis to specify a return sub-string, as per normal perl coding syntax. Only the first parenthesized sub-string will be returned.
- **service-definition** - This is a special match type that looks for subordinate field structures within the current data, and then creates service definitions based on the subordinate data. This may be needed when devices have a variable number of services that cannot be easily represented by a fixed-field data structure. Refer to the online help for detailed information about this match type and the field syntax requirements.

  If a matching rule uses one of the string-comparison techniques described above (such as *contains* or *begins-with*) or uses a Perl regular expression, then you must also provide the matching string to be used for the comparison operation. If a matching rule does not use a string-comparison technique (as is the case with *use-value-as-is* and *is-null*), this attribute will not be available.

2.4 Rule Directives

The rule attribute stores the processing statement for the current matching rule (IE, it stores the *then* part of the if-then statement). The rule will only be executed if the match filter (as described in the preceding section) returns a *true* result.

> ⚠ The available rule directives will be determined by the matching filter in use. For example, the *is-null* matching type will only ever result in a true condition if the current field does not contain data. As such, it cannot be used with rule directives that use field data to populate a configuration entry, and can only be used with rule directives that allow the administrator to specify all needed values.

The rules types that are available for use are as follows:

Rule Types

- **Assign value to** - This rule simply assigns the current field value to the specified attribute. However, this field only works with attributes that can be used to uniquely identify the current host object. If this rule is selected, you must also choose the attribute type to be populated with the field data. The supported attribute types are host name, host alias, host address, primary record, host description, and host profile.
- **Convert dword and assign to** - This is a special rule that converts a *double-word* field value to regular text and assigns the resulting value to the specified attribute. This field only works with attributes that can be used to uniquely identify the current host object. If this rule is selected, you must also choose the attribute type to be populated with the field data. The supported attribute types are host name, host alias, host address, primary record, host description, and host profile.
- **Assign host profile** - This rule simply assigns a specified host profile to the currently selected host object. If this rule is selected, you must also choose the host profile to be used.
- **Assign host profile if undefined** - This is a two-part rule that first checks to see if the currently selected host object already has a host profile defined from another matching rule. If not, the current field value will be used for that attribute. If this rule is selected, you must also choose the host profile to be used.
- **Assign service** - This rule simply assigns a specified service entry to the currently selected host object, with the field data being used as the service entry name. If this rule is selected, you must also choose the service type to be used for the new service entry.
- **Resolve to parent** - This rule indicates that the current column contains the name of the current device' network parent. Note that this rule directive is only available if the *use-value-as-is* matching filter has been selected.
- **Assign object(s)** - This rule simply assigns a specified configuration attribute to the currently selected host object, and populates the attribute with a specified value. This rule is intended to be used in those cases where the input data does not provide the necessary attribute value, and as such is only available if the *is-null* matching filter has been selected (if you wish to use the field data for the value, use the *assign object if exists* rule instead). If this rule is selected, you must also choose the attribute type and value to use. The supported attribute types are configuration group, contact group, host group, parent, and service profile.
- **Assign object if exists** - This is a special two-part rule that first checks to see if the current field contains any data. If so, the specified configuration attribute will be created for the currently selected host object, with the field data being used for the attribute value. Note that this rule is only available when the *use-value-as-is* matching filter has been selected, since it is the only filter that is capable of providing an empty field without prior detection. If this rule is selected, you must also choose a configuration attribute type to be created. The supported attribute types are configuration group, contact group, host group, host profile, parent, service profile, and service entry.
- **Assign value if undefined** - This is a two-part rule that first checks to see if the currently selected host object already has the specified configuration attribute defined from another matching rule. If not, the current field value will be used for that attribute. If this rule is selected, you must also choose the attribute type to be populated. The supported attribute types are host name, host alias, host address, primary record, host description, and host profile.
- **Add if not exists and assign object** - This is a special two-part rule that is intended to be used for creating certain types of global configuration objects on an as-needed basis. This rule first checks to see if a specified configuration object already exists with the same name as the field data. If the object does not exist, then a new configuration object will be created with that name. If this rule is selected,

you must also choose a configuration object type for comparison purposes. The supported configuration object types are configuration groups, contact groups, and host groups.
- **Discard record** - This rule simply causes the automation processor to discard the current record altogether. This rule is used by the default schema definitions to ignore comments in the input file by simply discarding all rows that begin with a # character.
- **Discard if match existing record** - This is a two-part rule that first checks to see if a host configuration object already exists with the same name as the field data. If so, the current record is simply discarded. This is useful for situations where you only want new devices to be added to the configuration database, and do not want existing host entries to be overwritten.
- **Skip column record** - This rule is used whenever a column contains multiple subordinate fields (as typically occurs when SNMP interfaces have been enumerated), and causes the automation processor to skip the current subordinate record in the current column.

By combining matching rules, it is possible to come up with some comprehensive processing rules. For example, the default schema definitions include three matching rules that are associated with the first column of data that perform the following steps:

1. If the field data begins with a # character, discard the record since it is a comment and not a record.
2. Use the field data for the host name attribute of a new record.
3. If the field data matches a host entry in the configuration database already, discard the record since it is a duplicate.

Another good example of a column with multiple rules can be found in the default schema definition under the *Description* column.

3.0 Managing Automation Schema Definitions

Automation schema definitions are managed with the Automation console screen. To access the Automation console screen, select the Auto Discovery menu item from the main menu, and then select the Automation menu item in the top menu bar.

The default Automation console screen is shown. As can be seen from that example, GroundWork Monitor provides a default schema definition called *GroundWork-Discovery-Pro* which is created during installation.

To edit or execute an existing schema definition, select it from the list and click the *Next >>* button, which will result in the schema definition editor screen being loaded. From there, the schema definition can be modified or executed. These topics are discussed in the Editing Schema Definitions and Running Schema Definitions sections below.

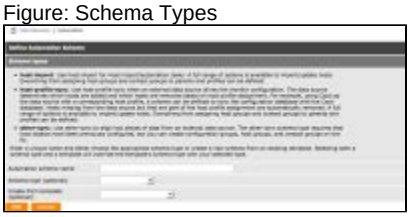
New discovery definitions can be created by clicking the *New Schema* button. This topic is discussed in the Creating Schema Definitions section below.

3.1 Creating Schema Definitions

To create a new schema definition, click the *New Schema* button on the automation console screen. A new screen like the one here will be shown.

Once the fields have been filled in to your satisfaction, click the *Add* button to continue. At this point the schema definition editor screen will be displayed, with the new schema definition values already loaded. If you do not wish to continue creating a new discovery definition, click the *Cancel* button to return to the main automation console screen.

Figure: Schema Types



Schema Fields

- **Name** - Enter a name for your new schema definition.
- **Schema type** - The drop-down list shows the three schema types that are available. One of the schema types must be selected to continue.
- **Create from template** - Optional Schema definitions can be saved as *templates* that allow them to be partially reused (see the discussion in the next section for more details).
  If you have already saved a schema definition as a template, you can reuse its options and variables by selecting it in the drop-down list. If you do not want to inherit any options from any other definitions, leave the drop-down list empty.

> ⚠ Choosing a schema template will cause all of the defined options to be inherited, including the schema type. As such, the schema type chosen in the drop-down list above will be disregarded.

3.2 Editing Schema Definitions

To edit the options and variables associated with a schema definition, select the schema definition in the automation console screen and then click the *Next >>* button. This will result in the schema definition editor screen being displayed with the schema definition values already loaded

(this screen will also be activated when a new schema definition is created, as described in the preceding section), similar to the screen shown below.

> ⚠ All of the schema types share a common layout, except for a couple of fields. In particular, all of the schema types share common settings and options for attributes such as field separator and rule definitions. However, the host-import and host-profile-sync schema types has an additional field for indicating whether or not the device name and IP address should be determined automatically, and another field for defining the default host profile that should be used for undefined entries. Meanwhile, the other-sync schema type has a separate field for defining the attribute that should be used as the primary synchronization key.

The figure shows the automation schema editor for the host-import schema type, using the values from the default *GroundWork-Discovery-Pro* schema definition.

Figure: Host Import Schema

The figure below shows the automation schema editor for the other-sync schema type, with no preloaded values. The options and fields in the schema definition editor screen are listed below.

Once you have finished making changes to the schema definition, click the Save button to make the changes permanent.

If you would like to create a template of the schema definition that can be applied to future definitions, click the *Save As Template* button. Templates are stored as textual XML files in the `/usr/local/groundwork/core/monarch/automation/templates` directory on the GroundWork Monitor server. Template files can be copied into the same directory on another GroundWork Monitor server and will be immediately usable on that system.

If you want to initiate a automation process using the current values but without making the changes permanent, click the *Process Records* button. If you want to cancel your changes and return to the main automation console screen, click the *Close* button.

Figure: Other Schema

Schema Definition Options and Fields

- **Smart Names (host-import and host-profile-sync only)** - GroundWork Monitor's configuration database stores multiple host-specific attributes, including a device short name, a device alias name (which is typically the fully-qualified domain name), and the IP address of the device. If the schema definition uses the host-import or host-profile-sync schema types, and if the input file does not provide all of these attributes, then GroundWork Monitor can attempt to detect the data automatically with a variety of lookup queries. Note that the input data must provide at least one of these three attributes in order for this process to work correctly. Also note that this field does not exist in the schema definition editor screen that is used with other-sync schema types, since the relevant attributes are not modified by that schema type. Due to the overhead required in performing these lookups, this option is disabled by default, although it is enabled in the *GroundWork-Discovery-Pro* schema definition.
- **Primary Sync Object (other-sync only)** - In order for the other-sync schema type behavior to operate correctly, it must be able to match input data to existing entries. This drop-down list shows the fields that are usable for this purpose. By default, entries are matched by the host-specific short names, but they can also be matched by broad-based categories such as host group or contact group if those configuration entries need to be modified.

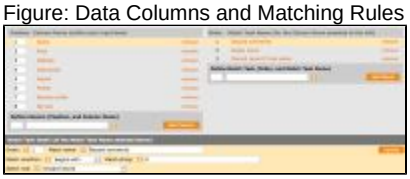
- **Data Source** - This field specifies the path to the input file that will be read and parsed for the automation process. Note that this field is not used for automation processes that are started by the discovery process. Instead, those jobs use a filename that is generated dynamically from the name of the current discovery definition, with the input file being stored in the `/usr/local/groundwork/automation/data` directory (this allows multiple discovery jobs to keep their discovery data somewhat independent of each other).
- **Delimiter** - This field is used to specify the delimiter that is used to separate input fields from each other, when multiple fields are provided in the input data. As stated above, each row of data in the input file may only specify a single record, but each record may contain multiple fields if they have a common delimiter. Users have two choices for entering data here. First, the drop-down list provides a convenient way to choose from several common field delimiter characters. Meanwhile, the edit box allows administrators to specify their own character sequences. The check box between the drop-down list and edit box is used to indicate that the latter should be used, but is not selected by default. In the case of the default *GroundWork-Discovery-Pro* schema definition, a pair of semi-colon characters is used as the field delimiter.
- **Data Columns (and subsequent fields)** - This section of the schema definition editor screen is used to define columns of data in the input file, and also used to define the matching rules that are associated with each column. These topics are explored in detail in the Matching Rules section above, while instructions for using these options are provided in the Managing Data Columns and Matching Rules section below.

4.0 Managing Data Columns and Matching Rules

As was discussed earlier, the automation processor reads in lines of data from the input file, and then extracts the individual fields of data from the input by looking for a sequence of user-defined separator characters. These fields are then mapped to a temporary grid of rows and columns, with one record per row and one field per column. Each column has one or more matching rules that are executed as the field data is extracted, with the matching rules ultimately determining what will become of the data (IE, will the data be mapped to a field in the GroundWork Monitor configuration database, or will it be discarded, and so forth).

In order for this process to work, administrators must define the columns of data that are found in the input file at hand, and must also define the matching rules that dictate how each column will be processed by the automation subsystem. These tasks are performed in the schema definition editor screen, which contains a *Data column* section for just this purpose. The figure here shows this portion of the screen for the default *GroundWork-Discovery-Pro* schema definition.

Figure: Data Columns and Matching Rules

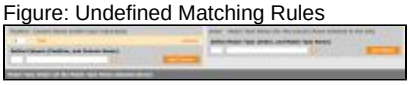


The next figure below shows this portion of the screen for a new schema definition that does not yet have any columns or matching rules defined.

> ⚠ Data columns must be defined before matching rules can be assigned to the column. This is illustrated by the second figure above, which shows no columns, and therefore has no facilities for creating or editing the matching rules.

When multiple column definitions or matching rules exist, the currently selected definition is highlighted with a dark yellow background. When only one definition exists, it is always selected and therefore always highlighted.

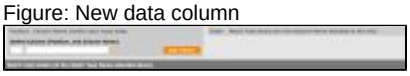
Figure: Undefined Matching Rules



4.1 Managing Data Columns

To create a new data column mapping, fill in the fields in the *Define Column* portion of the screen to suit your requirements.

Figure: New data column



If you wish to change the field position of a data column definition, overwrite the numeric value in the Position field and click the *Save* button at the top of the schema definition editor screen.

If you wish to rename a data column, it must be deleted and recreated. However, this process will also result in the deletion of the matching rules that are associated with the data column.

To delete a data column definition, click the *remove* hyperlink next to its entry.

Once the fields have been filled in to your satisfaction, click the *Add Column* button to continue. At this point the schema definition editor screen will be redrawn, the new data column will be shown and automatically selected, and facilities for creating matching rules for that column will also be enabled. This is illustrated by the figure below, which shows a new schema definition with a single *Test Column* that was just created:

Define Column Fields

- **Position** - Enter a numeric value that reflects the sequence number of the data field in the input file (the first field of data is column number *1* and so forth).
- **Column name** - Enter a descriptive name for the data column.

4.2 Managing Matching Rules

Figure: New matching rule

To create a new matching rule, click on the name of the data column, and then fill in the fields in the *Define Match* portion of the screen to suit your requirements.

> ⚠ The process above only creates the matching rule placeholder, but the matching rule semantics have not yet been defined. In order to set these parameters, the matching rule must be edited.

To edit a matching rule, click its hyperlinked name. This will result in an additional *Match Detail* dialog box being shown that allows you to change all of attributes of the matching rule, including its priority order, name, and the rule semantics. Any previously defined options for that rule will be loaded when the dialog is drawn.

> ⚠ The number of fields in this dialog will vary according to the matching filter and rule directives that are used by the currently selected matching rule.

Once the fields have been filled in to your satisfaction, click the *Add Match* button to continue. At this point the schema definition editor screen will be redrawn, and the new matching rule will be shown and automatically selected. This is illustrated by the figure below, which shows a new schema definition with a single *Test Column* and a single *Test Rule* that was just created.

Matching Rules

- **Order** - Enter a numeric value for the matching rule priority. Multiple matching rules can be associated with a data column, and each rule will be executed in sequence according to their priority order value (matching rule *1* will be executed before matching rule *2* and so forth).
- **Match task name** - Enter a descriptive name for the matching rule.

The figure here shows this dialog with a new (as-yet-undefined) matching rule.

Meanwhile, the figure below shows another example of this dialog box with additional fields that reflect the matching filters and rule directives that have been selected.

Once the fields have been filled in to your satisfaction, click the *Update* button to continue. At this point the schema definition editor screen will be redrawn, and the modified matching rule will be shown.

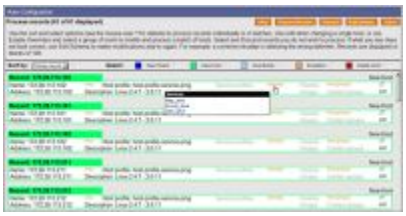
To delete a matching rule, click the *remove* hyperlink next to its entry.

Match Detail Fields

- **Order** - Enter a numeric value for the matching rule priority.
- **Name** - Enter a descriptive name for the matching rule.
- **Match** - This drop-down list contains the available matching filter types. The matching filter types control the conditional branching portion of the matching rule, and also determine the rule directives that are available in the *Rule* field below. The available match types are discussed in detail in the Match Filters section above.
- **Match String** - If the match field above uses a string-comparison filter (such as *exact* or *contains*), a new field will appear that allows you to enter the string value that you want to use for the match. This field is not displayed if a string-comparison matching filter is not used.

- **Rule** - This drop-down list contains the available rule directives, as determined by the matching filter type selected in the field above. The rules determine what will actually happen with the field data when it is read into the column (IE, whether it will be mapped to a field, discarded, or something else).
- **Object** - If the rule directive allows a value to be defined to a configuration object (regardless of the object type or the value being defined), the Match Detail dialog will provide an *Object* drop-down list that allows you to choose the object to be changed. This field is not displayed if the rule directive does not modify a configuration object.
- **Object-specific value** - If you have chosen a rule directive that allows an object to be set to a predefined value, a multiple-item list box will be shown that contains all of the known values for that object. In the example shown above, the administrator has chosen a rule directive that allows the *host group* attribute for the current device to be set to a predefined host group, so an additional dialog element is displayed that allows the administrator to choose from one of the known host groups.

5.0 Running Automation Processes



A schema definition can be executed as part of a discovery process, or as an independent operation. In the former case, the automation process may be initiated from the discovery console screen or the discovery definition editor screen as described in the Running Discovery Processes section above. In the latter case, automation processes can only be executed by clicking the *Process Records* button in the schema definition editor screen.

When the automation process begins, the contents of the input file will be parsed as described in the Automation Process section above. Once this process completes, you will be presented with a summary screen that shows all of the discovered configuration objects and their major attributes. The figure below shows an example of what this screen can look like, using the sample Multi-Line-Data schema definition and input file:

This screen provides a summary view of the configuration entries that have been detected in the input file, and also provides mechanisms for editing the schema and individual entries, and options to select subsets of the entries for further processing.

The center of the screen shows a scrollable child window that contains the configuration objects that are pending some kind of action (such as a new record waiting to be added, or an existing record waiting to be updated or deleted). Existing entries in the configuration database that are not being modified will not be shown. Similarly, entries that previously had a pending operation but have already been processed or discarded will no longer be pending, and are removed from the summary screen when those changes occur.

Each entry in the summary screen has a checkbox to the right which allows you to select an individual record for manipulation. If you wish to work with multiple entries simultaneously, you can click each entry's individual checkbox, or you can click the *Check All* button in the bottom right corner of the screen to select all of the entries. You can also click the hypertext color names in the color legend at the top of the screen to select entries of that type (such as selecting all host records that are flagged for deletion, which is sometimes useful for avoiding the destructive effects of a full synchronization).

Once the desired entries have been selected, click the *Process Records* at the top of the screen to integrate those records with the GroundWork Monitor configuration database, or click the *Discard* button to drop those records from the pending list. If you wish to change the schema definition that was used to generate the entries, click the *Edit Schema* button at the top of the screen, and you will be returned to the schema editor screen (described in the Editing Schema Definitions section above). If you wish to cancel the automation process entirely, click the *Close* button to return to the main menu.

Each entry in the summary screen has a color that reflects its condition. For example, host records for devices that do not already exist in the configuration database have a green coloring, while host records for devices that do already exist have a light blue coloring (by default, existing devices are not shown in the summary list, since the default schema definitions drop duplicate host records). The color legend at the top of the window shows these colors and their meanings.

Host records in the child window can be resorted according to the primary key, host name, host alias, or host address by choosing the desired sorting method from the *Sort by* drop-down list.

The major attributes for each entry can be reviewed by moving your mouse over the hypertext elements in each row. For example, moving your mouse over the *Alias* hyperlink in the *bern* entry will show that the device's alias has been detected as *bern.alps.com*, while moving your mouse over the *Services* hyperlink for the *router-1* entry will show all of the interface-specific service entries that have been discovered for that device.

You can change the attributes of multiple records simultaneously by clicking the *Enable Overrides* button in the bottom left corner of the screen. This will cause a new dialog area to be displayed at the bottom of the screen, similar to the this image.

This screen allows you to choose the attribute that you want to override, and the attribute value(s) that you want to force onto the selected records. You can also choose whether to completely replace the discovered values (*Replace*), or if you want to append the selected value(s) to the discovered values (*Merge*). Note that only the attributes that have a selected checkbox next to their name will be modified.

Once the desired modifications have been made, select one or more entries from the summary list, and click the *Process Records* button at the top of the screen to process the selected entries with the requested changes. If you decide that you do not wish to force overrides on any records at this time, click the *Disable Overrides* button to close the dialog.

The individual attributes for an entry can be edited by clicking the *edit* hyperlink to the right of the entry. Clicking this link will cause a record editor screen to be displayed with the major attributes of

the selected record loaded. This figure shows the editor screen for the *zurich* device entry.

As can be seen, the record editor screen allows the administrator to override any of the major configuration attributes that have been assigned to the entry. This includes the discovered services and the arguments associated with each service definition (selecting a service entry will allow you to edit the arguments).

Note that any changes you make will only be recognized if commit the change by clicking the *Process Record* button. You can also discard the record, or cancel your changes (resulting in the loss of your edits).