



# **Discovery Genie 3.1**

## Active Discovery Rules Reference Guide



**Jilroy Software LTD 2010**

Jilroy Software makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Jilroy Software shall not be held liable for errors contained herein or direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of this material.

All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Jilroy Software. The information contained in this document is subject to change without notice.

Copyright © 2000-2010 Jilroy Software. All rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

Microsoft® is a US registered trademark of Microsoft Corporation.

Windows® and MS-Windows® are US registered trademarks of Microsoft Corporation.

Pentium® is a US registered trademark of Intel Corporation.

UNIX® is a registered trademark of The Open Group.

All other company and product names may be trademarks or registered trademarks of their respective owners.

## Table of Contents

<b>Preface.....</b>	<b>10</b>
<b>Introduction.....</b>	<b>12</b>
<b>Defining Discovery Rules.....</b>	<b>15</b>
General Description.....	16
Discovery Items Hierarchy.....	16
A schematic view of Discovery Items relationship.....	17
Discovery rules definition GUI.....	18
The wizard general functionality.....	18
Discovery Item Object.....	19
Discovery items hierarchy.....	19
Discovery Items parameters.....	19
Flexible targets of the discovery process.....	19
Update mode.....	19
Conditional discovery.....	19
Non update discovery.....	20
Discovery Item properties panel.....	20
Discovery Item properties.....	21
Name.....	21
Description.....	21
Path.....	21
Table Name.....	21
Table Description.....	21
Type.....	21
isRoot.....	22
ParentNameList.....	22
SkipResults.....	22
UpdateOnChangeOnly.....	22
UpdateOnChangeOnlyFieldList.....	22
Delete command.....	22
ID.....	22
Discovery Item XML format.....	23
Condition.....	24
Fields Compare Condition.....	24
Key.....	25
Value.....	25

Function.....	25
Fields Compare Condition XML format.....	25
Dates Compare Condition.....	26
Date Pattern.....	27
Time Pattern.....	27
The logic of using the Date and Time pattern.....	28
Discovery Field Item.....	28
Data parsers.....	29
Common Field Item properties.....	29
Method.....	29
Field Type.....	30
Table Field Name.....	30
Table Field Description.....	30
isKey.....	30
isTableKey.....	30
Unique field check – sql query.....	31
Final SQL query.....	31
Final SQL selected field name.....	31
Final SQL Key field name.....	31
Skip Protocol If Fails.....	31
ID.....	31
Direct Field Item.....	32
Value.....	33
Value Function.....	33
'Direct' Method XML format.....	33
A simple value XML.....	33
A Time value XML.....	34
AutoGenerateInt Field Item.....	36
'AutoGenerateInt' Method XML format.....	37
CSV Field item.....	38
URL.....	45
User.....	45
Password.....	45
SQL command.....	45
Query field name.....	45
Is First line a header.....	45
Key field name.....	45
'CSV' Method XML format.....	45

DiscoverIpRange Field item.....	47
Ip Range list.....	50
Timeout in ms.....	50
'DiscoverIpRange' Method XML format.....	50
Discover SNMP Community Field item.....	51
Ip Address.....	52
Community list.....	52
SNMP Version.....	52
Timeout in ms.....	52
Retry count.....	52
'DiscoverSNMPCommunity Method XML format.....	52
Is Telnet SSH Responding Field item.....	54
Protocol.....	54
Host name or IP.....	55
Port.....	55
User.....	55
Password.....	55
SSH authentication method.....	55
SSH PEM file.....	55
'IsTelnetSSHResponding Method XML format.....	55
LDAP Field item.....	57
Ldap URL.....	57
User.....	58
Password.....	58
LDAP Source.....	58
LDAP Query.....	58
LDAP Method XML format.....	58
Open Tcp Ports Field item.....	60
IP Address.....	60
Port Range list.....	61
Timeout in milliseconds.....	61
Max Concurrent socket checks.....	61
Open TCP Ports Method XML format.....	61
SNMP Field item.....	63
IP Address.....	63
Community.....	64
SNMP Version.....	64
Timeout in milliseconds.....	64

Retry count.....	64
Max PDU size.....	64
OID.....	64
OID description.....	64
Can data be in hex format.....	64
Get Key value of SNMP reply.....	64
Skip protocol for node in case of an Error.....	64
SNMP Method XML format.....	65
Registry Field item.....	67
Root Key.....	67
WMI Path.....	68
Query type.....	68
User.....	68
Password.....	68
IP.....	68
Field name.....	68
Key field name.....	68
Registry Method XML format.....	69
SQL Field item.....	70
SQL Driver name.....	70
SQL Connection string.....	71
User.....	71
Password.....	71
SQL command.....	71
Query field name.....	71
Table Key field.....	71
SQL Method XML format.....	71
Retry count.....	72
Telnet / SSH Field item.....	73
Protocol.....	74
Host name or IP.....	74
Port.....	74
User.....	74
Password.....	74
SSH authentication method.....	74
SSH PEM file.....	74
Script file name.....	74
Script data.....	74

Script parameters.....	74
Control commands within the script data.....	75
Prompt.....	75
RegexPrompt.....	75
CommandTimeout.....	75
FailOnError.....	75
Telnet vs. SSH handling of the script control commands.....	75
'TelnetSSH' Method XML format.....	76
System Command Field item.....	78
Script file name.....	79
Script data.....	79
Script Parameters.....	79
'SystemCommand Method XML format.....	79
WMI Field item.....	81
Ip Address.....	89
Path.....	89
User.....	90
Password.....	90
Query command.....	90
Field Name.....	90
Table Key Field.....	90
'WMI' Method XML format.....	90
Async Ping Field item.....	92
Ip List Field Name.....	93
Timeout in ms.....	93
Retry count.....	93
Window Size.....	93
Requests count sent in one second.....	93
Async Discover SNMP Community Field item.....	94
Ip List Field Name.....	95
Community list.....	95
Timeout in ms.....	95
Retry count.....	95
Window Size.....	95
Requests count sent in one second.....	95
Async SNMP Field item.....	96
Ip List Field Name.....	97
Community Field Name.....	97

Timeout in ms.....	97
Retry count.....	97
Window Size.....	97
Requests count sent in one second.....	97
Async Open Ports Field item.....	98
Ip List Field Name.....	99
Port Number.....	99
Timeout in ms.....	99
Max concurrent socket checks.....	99
<b>The Data parser.....</b>	<b>100</b>
The Data parser entry.....	101
The Paragraph parser entry.....	101
The Table parser entry.....	102
The Line parser entry.....	102
The Field parser entry.....	103
The Regular expression Field Array parser entry.....	103
The Head Position entry.....	104
The Tail Position entry.....	104
The locaters.....	104
The text locator.....	104
Start Search from.....	105
Take Instance number.....	105
The data to look for.....	105
Type of search.....	105
Action when not found.....	105
The line locator.....	106
Start Search from.....	106
Take Instance number.....	106
Action when not found.....	107
The word locator.....	107
Start Search from.....	107
Take Instance number.....	107
Action when not found.....	107
The offset locator.....	107
Start Search from.....	107
Take Instance number.....	108
Action when not found.....	108
<b>Final Page.....</b>	<b>109</b>



More Information..... 109

Contact Us..... 109

Sales..... 109

Product Management..... 109

# Preface

---

Welcome to Discovery Genie 3.1 The Active Inventory & Discovery engine of Jilroy Software. This chapter provides an introduction to the structure and assumptions of this guide.

## The Purpose of This Guide

This guide contains information needed to use Discovery Genie 3.1 from Jilroy Software efficiently and effectively.

## Who Should Use This Guide

This guide is intended for programmers and the advanced users of the products using the discovery engine as their base tool.

## Organization of This Guide

This guide is structured to reflect the following conceptual divisions:

- Preface – A description of the guide purpose, intended audience, organization, and conventions.
- Introduction – A general description of Discovery Genie 3.1 and its goals.
- Defining discovery rules – A description of the how to define discovery rules for the product.
- Final Page – Information about contacting Jilroy Software.

## Conventions

The manual uses the following conventions:

- Names of dialog boxes, windows, and unnamed screen areas are displayed in *italics*.
- Names of buttons, tabs, check-boxes, and other screen elements are displayed in **bold**. For example, click **OK** or type the **Start date**.
- **This font** is used for text that you enter.
- `This font` is used for code, directory names, file names, and system activity.
- UPPERCASE is used for keys and acronyms.
- Steps that involve two or more selections from a menu may be presented as a combination of selections separated by an > angled bracket.

For example, when you see **File > New**, click the **File** menu on the menu bar. This will open a drop-down menu. Then select the **New** command.

- Cross-references are underlined. For example, see Chapter 2.
- Hyperlinks are underlined and blue.
- The ⓘ symbol signifies notes, which are used to provide extra or special information regarding the preceding topic.

- The *Italic* font style is used to *emphasize* words and phrases in special cases.

# Introduction

---

Discovery Genie 3.1 from Jilroy Software is a very powerful generic discovery tool. It can be used to perform discovery operations and perform actions on any subject: Possible uses are:

1. Network Discovery
2. Servers inventory
3. Business inventory
4. Active Inventory:
  - Compliance
  - Action automation

The product currently supports inputs from the following sources:

1. ICMP(Ping)
2. SNMP v1/v2
3. Telnet protocol
4. SSH protocol
5. SQL based data sources
6. CSV data sources
7. WMI
8. Registry
9. HTTP
10. Open TCP port
11. LDAP

The product also supports inputs from certain sources used for high speed discovery. The problem these input sources face are from very large scale organizations, when discovery processes done synchronously are not fast enough. For that the product for certain inputs supports asynchronous sources.

Currently we support:

1. Asynchronous ping
2. Asynchronous snmp community discovery
3. Asynchronous snmp
4. Asynchronous Open port scanning

and more if needed. Please contact us in order to find which data-source is ready for testing.

Using Discovery Rules you define to Discovery Genie 3.1, you can determine what will be the structure of the output inventory database, and from where the input data will be taken.

You can also determine which actions will be performed and which commands executed based on the currently discovered data.

In some complex organizations, there are remote sites that have the same internal IP addresses and their discovery from the center is not possible. With this platform you can perform remote discovery operations on such sites, even across firewall. The discovery policy is still set in the center, and the accumulated data collected from the remote site, is put in the central database.

A full description on how to define the Discovery rules will be supplied in a later chapter.

## Advantages of the product

- The product is one of the most flexible products existing in the market today. You as the customer have control on every thing discovered. You can alter and add discovery rules as you wish in a very easy way using a GUI.
- Its output is redirected into a database which can be accessed using jdbc
- It can be ran as a full Java based solution and as such can run on any java supporting platform from Unix platforms (Linux/sun/hpux) to Windows and others. (It has some capabilities that require JNI and running on specific platforms, such as Direct WMI, using Microsoft com.)
- You can also alter and customize the collected fields on the discovered items using configuration files, so now you can tailor the discovery process to your exact needs in programmatic tools too.
- The product supports high speed discovery processes used form very large organizations, when regular discovery process take too long.
- The product supports remote discovery processes that enable discovering from the center, what is found in sites located beyond firewalls, and even having the same IP addresses as other sites.

Jilroy Software encourages you to use the product and extend it, by defining your own discovery rules.

If you have any questions and recommendations about how to use or how to improve the product you are most welcome to send us your thoughts to [support@jilroy.com](mailto:support@jilroy.com)

If you need us to tailor the product to your exact needs we are offering this kind of service for reasonable fees.

# Defining Discovery Rules

---

The product's purpose is to supply an Active Inventory tool. It is built to fill Inventory databases, when the content of the inventory can be any thing, that can be extracted using the plug-ins supplied by the product.

It can also perform any action based on the discovered data, making it a very powerful automation and compliance tool.

The product was built for maximum flexibility and can be extended easily to additional sources.

Currently the following sources are supported:

1. ICMP (Ping)
2. SNMP
3. SQL (JDBC) drivers
4. Telnet protocol
5. SSH protocol
6. CSV (file & URL based csv files)
7. HTTP protocol
8. WMI queries
9. Registry queries
10. Open TCP ports discovery

There are 2 methods for defining discovery rules.

- A simple GUI wizard.
- Editing or creating discovery rules in a programmatic way

The discovery language is XML.

## General Description

An Inventory process is composed of one or more discovery items which in turn define the contents of tables in the inventory database. The discovery items format a structure of a tree when a parent discovery item can have multiple child discovery item.

Each discovery item defines which table does it process and which fields in that table will it fill.

There are 2 Major types of discovery elements that can be defined:

1. Discovery Item - which define single lines or groups of lines in their target inventory destination table.
2. Discovery Fields - Which are a sub-component of the discovery items and define the contents of the fields in the lines added to the inventory table, by the discovery item. There exist also commands based discovery fields that actually perform commands, either locally or using some of the supplied protocols, for example telnet & ssh.

Our products which are using this inventory technology come with prepared discovery rules for multiple purposes.

Please look at our site for available discovery rules.

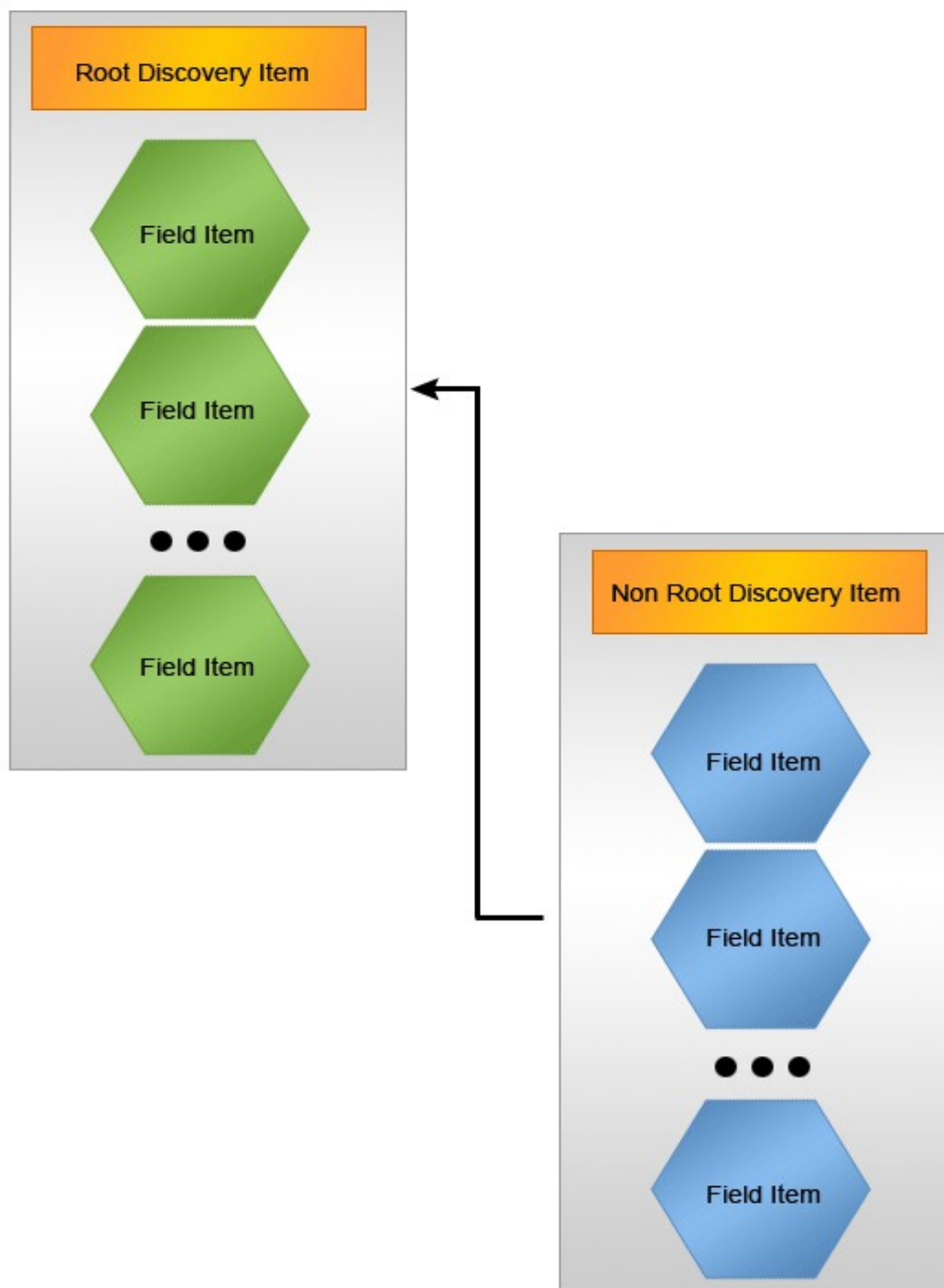
## Discovery Items Hierarchy

Discovery items are the building blocks of creating an active inventory. When we run an active inventory job, we start with a root discovery item which may point to other discovery items.

In this section we explain schematically the relationships between discovery items and between discovery items and their field items



## A schematic view of Discovery Items relationship

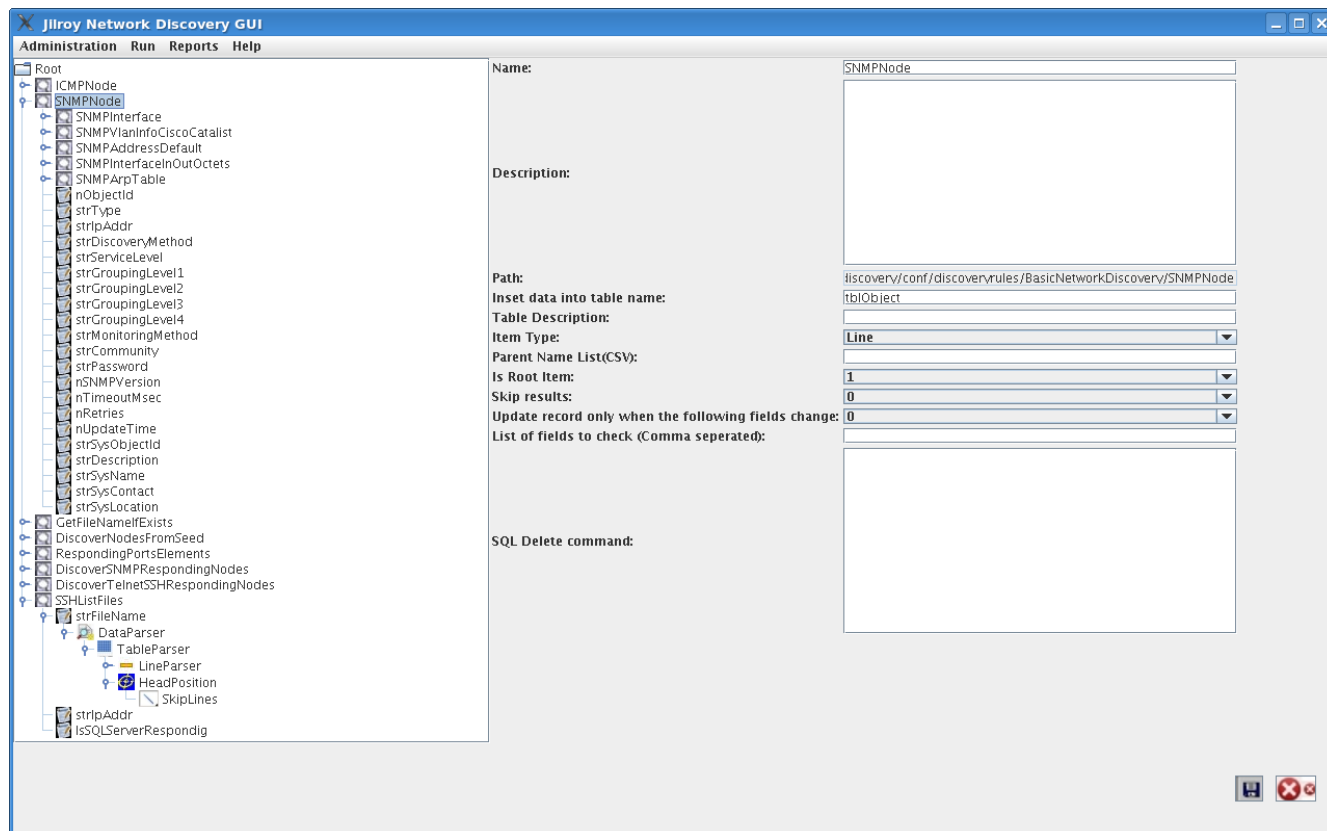


## Discovery rules definition GUI

To enter the discovery rules definition panel you should select:

**Administration->Discovery rules->Discovery rules definitions**

The panel opened is:



This panel is a wizard used to define discovery item sequences.

The panel is divided into 2 sections:

- A tree view of the discovery items and their hierarchical structure
- A properties pane, that lists the specific parameters of a given discovery item or field.

### The wizard general functionality

- You can right click on every item and then a pop-up menu with possible actions that can be performed on the selected node will appear.

- You can click on the name of a tree item to see its properties pane
- You can click the open/close icon near the tree element to open its sub-tree.

## Discovery Item Object

The Discovery Item Object, is the main container of instructions on how to discover rows with in an inventory target table. It can define rules about the content of a single row or of a related set of rows in the target table.

### *Discovery items hierarchy*

The discovery process starts from a 'root' discovery Item, and then proceeds to all its children. Each discovery Item has and can use all the information generated by its parent chain when extracting data that should be inserted into its target inventory table rows.

### Discovery Items parameters

A discovery item, can use parameters that are passed to it by its parent, and in the case of a 'root' discovery item, by the launcher of the discovery process. The parameters are string based, and contain a key and a value. The parameters are referred in the child discovery item in the following format: the key string is wrapped in the format of @key\_name@ and is put in a location of a specific property. In run time the key value is replaced by the actual value of the parameter and is used to execute the instructions on how to find the value of the given table fields set by the discovery item.

All fields selected by a parent discovery item, can be accessed as parameters by its children.

### Flexible targets of the discovery process

Since the discovery application is a very flexible and generic, you can discover any objects of any kind, there is no limitations to the location of the output of the discovery process, and within a given discovery item, the user can direct the output to any given table. The only limitation is that a given Discovery Item will update a single table within the inventory database.

The table name and the fields updated are set using the configuration phase of the discovery rules.

### Update mode

A discovery Item does not only have to create new records. It has the capability of updating existing records in a given table. In the update mode, there should be definitions in the level of the discovery item, that specify which field(s) define the key of the entered line. The Discovery process supports keys composed from multiple fields.

### Conditional discovery

Some times discovery items should only be processed if a certain condition occurs. For example we only want to do a WMI discovery only to windows based nodes. Within the discovery item definition you can specify the conditions that should occur for the discovery item to be processed.

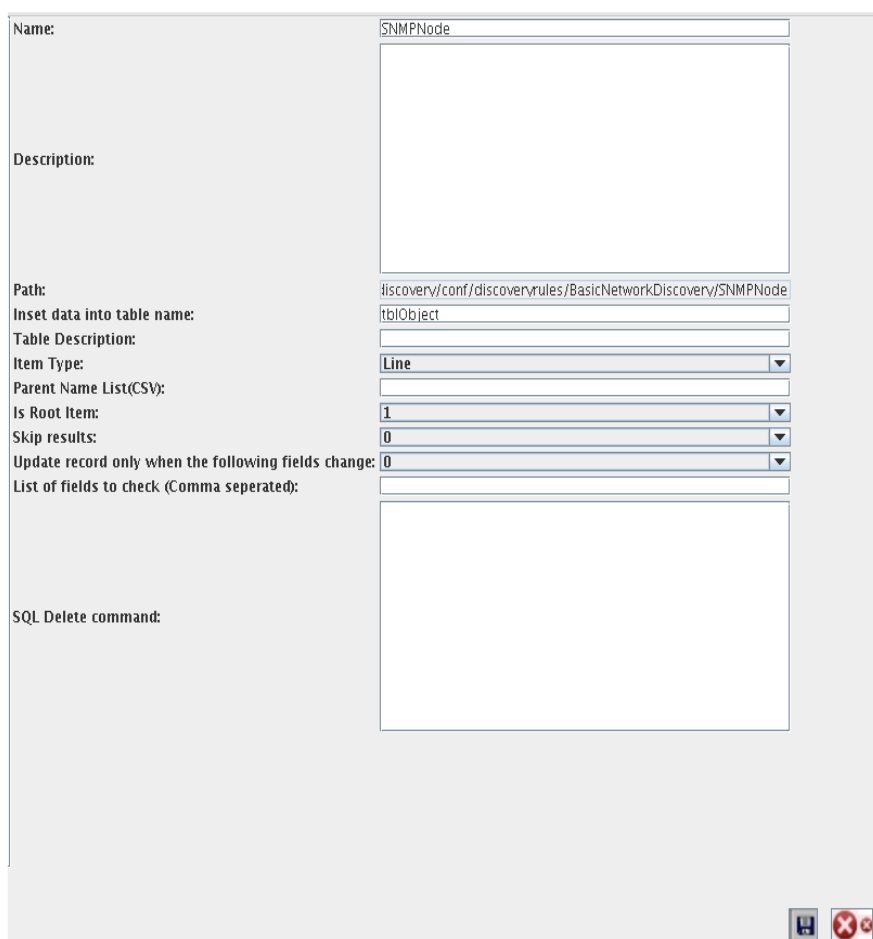
## Non update discovery

Some times the user will want to create a discovery item, that will not change the discovery database. This is usually done in order to extract data that will be used by child discovery items.

The user can specify that a given discovery item will not change the database.

## Discovery Item properties panel

The following panel displays the properties that can be set in the definition of a Discovery Item.



The screenshot shows the 'Discovery Item properties panel' with the following fields and values:

- Name: SNMPNode
- Description: (empty text area)
- Path: /discovery/conf/discoveryrules/BasicNetworkDiscovery/SNMPNode
- Inset data into table name: tblObject
- Table Description: (empty text area)
- Item Type: Line (dropdown menu)
- Parent Name List(CSV): (empty text area)
- Is Root Item: 1 (dropdown menu)
- Skip results: 0 (dropdown menu)
- Update record only when the following fields change: 0 (dropdown menu)
- List of fields to check (Comma seperated): (empty text area)
- SQL Delete command: (empty text area)

At the bottom right of the panel, there are two icons: a save icon (floppy disk) and a close icon (red X).

You can either save the changes you do or close the properties screen without saving.

In the following section we will explain the use of each property in the definition of a discovery

item, and the structure of the XML that constructs the Discovery Item XML.

## Discovery Item properties

The following section will list the properties used to define a discovery item.

### ***Name***

This property represents the name of the discovery item. It must be unique to all discovery items. When prompted from the GUI, the name can include a path name relative to the [INSTALLDIR]\conf\discoveryrules\ directory if the user wants to create an hierarchy of discovery items located in directories.

The name selected will be the name of the XML that will hold the file name.

Valid values: a valid file name

### ***Description***

This property holds a description of the discovery item. This description will appear in the reports that list the discovery items information.

Valid values: a string, not tampering with the XML format.

### ***Path***

A display field in the property panel that lists the location of the discovery item.

### ***Table Name***

This is the name of the table that will hold the output generated by this discovery item. The product will create the table if it does not exist.

### ***Table Description***

This is a description of the created table used for documentation and in reports listing the tables created by the defined discovery rules.

### ***Type***

A discovery item can update a line (row) or a set of lines (rows) (table) in the target Database. For example: think of the interface table in a router. a single query can return the value for all the interfaces, and as such, the result defines a set of lines (a table). An example for a single line query, is the query for the system parameters of a given node (SysName, SysObjectId, SysDescription,...). The result is a single Line. and as such it fits into single row.

In this field you define the type of the discovery item.

Valid values are: Line, Table

***isRoot***

A root discovery item, is usually the start of the discovery process. In this field we specify if this discovery item is a root item. All root discovery items will be displayed in the discovery rules definition GUI as root items even if they are children to other discovery items too.

Valid values are: 0, 1

***ParentNameList***

The names of the parent discovery items of the current item. This is a CSV list (Comma separated list)

Valid values: same as the 'Name' tag.

***SkipResults***

This field determines whether the output of this discovery item will change the discovery database, or will only be used as a way to pass parameters to its children.

***UpdateOnChangeOnly***

This field determines if we will update a record only when there is a specific field change that occurred. For example sometimes we will want to know when a record changes (i.e. a specific value in the record has changed), in this case we will use this option together with the UpdateOnChangeOnlyFieldList tag.

Valid values are (0|1).

***UpdateOnChangeOnlyFieldList***

This field contains a csv (Comma Separated Vector) of fields that must be changed for the update to the record to occur. This option comes together with the UpdateOnChangeOnly tag.

Valid values are fields in the updated table that must be changed for the update to be done.

***Delete command***

This SQL command is run prior to the discovery process in order to delete records already discovered when a specific condition occurs.

***ID***

This field only appears in the XML of the discovery item, and contains a unique ID between all discovery item elements. It is automatically generated if there none existing.

## Discovery Item XML format

The following section will list the tags available in the discovery rules item.

```
<DiscoveryItem>

    <strId>cdd9beac-92d1-4354-b77c-e8408e47b147</strId>

    <strName>SNMPNode</strName>

    <strDescription></strDescription>

    <strTableName>tblObject</strTableName>

    <strTableDescription></strTableDescription>

    <strParentNameList></strParentNameList>

    <fIsRoot>1</fIsRoot>

    <fSkipResults>0</fSkipResults>

    <fPossibleUpdateLine>0</fPossibleUpdateLine>

    <strType>Line</strType>

    <strSQLDeleteCommand></strSQLDeleteCommand>

    <strStartupParametersFilePath></strStartupParametersFilePath>

    <strUpdateOnChangeFieldList></strUpdateOnChangeFieldList>
    <Condition>

        ...

    </Condition>

    <FieldItem>

        ...

    </FieldItem>

    ...

    <FieldItem>

        ...

    </FieldItem>
```

```
</DiscoveryItem>
```

## Condition

The condition object is an object which appears within the Discovery Item definitions. It defines what are the conditions that should occur for this discovery item to be executed.

There could be zero or more conditions, and the relationship between them is an 'AND' relationship.

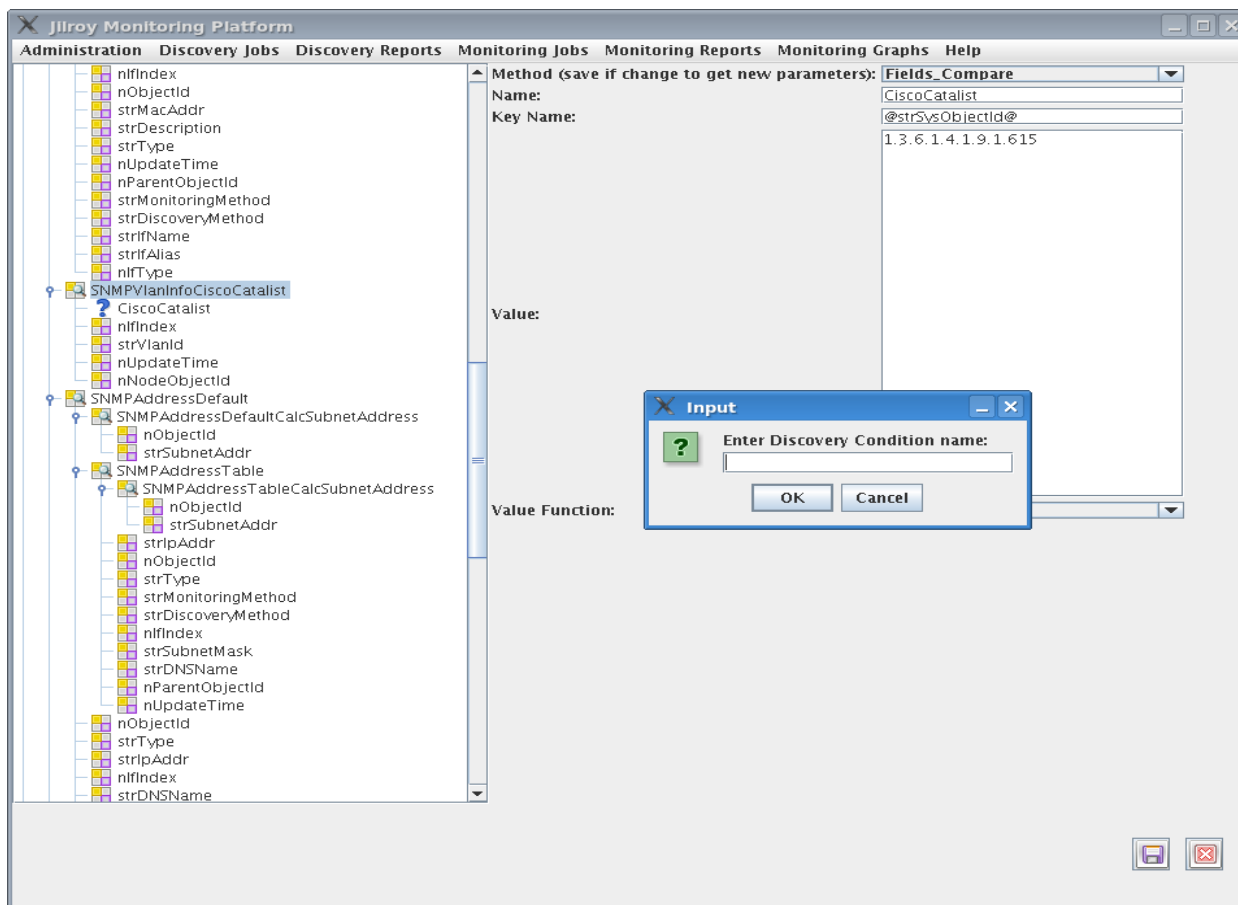
There are several types of Conditions currently defined and new are added as time passes.

The current Condition methods are:

- Fields Compare conditions
- Date Compare conditions

### ***Fields Compare Condition***

This condition allows comparing field values extracted to each other or to specific values.





The following is a list of available parameters within the condition method:

- Key
- Value
- Function

### **Key**

A name of a parameter whose value should be replaced, and then evaluated

### **Value**

The value used in the function, against the value of the parameter. This can also be a parameter too.

### **Function**

The name of the function to execute on the supplied parameters.

The currently supported functions are:

- KeyContainValue – returns true if the content of the parameter supplied contains the value supplied in the value key.
- KeyStartWithValue – returns true if the content of the parameter supplied starts with the value supplied in the value key.
- KeyEqualValue – returns true if the content of the parameter supplied is equal to the value supplied in the value key

## **Fields Compare Condition XML format**

The following section will list the tags available in the Condition item.

```
<Condition>

    <strKey>@strSysObjectId@</strKey>

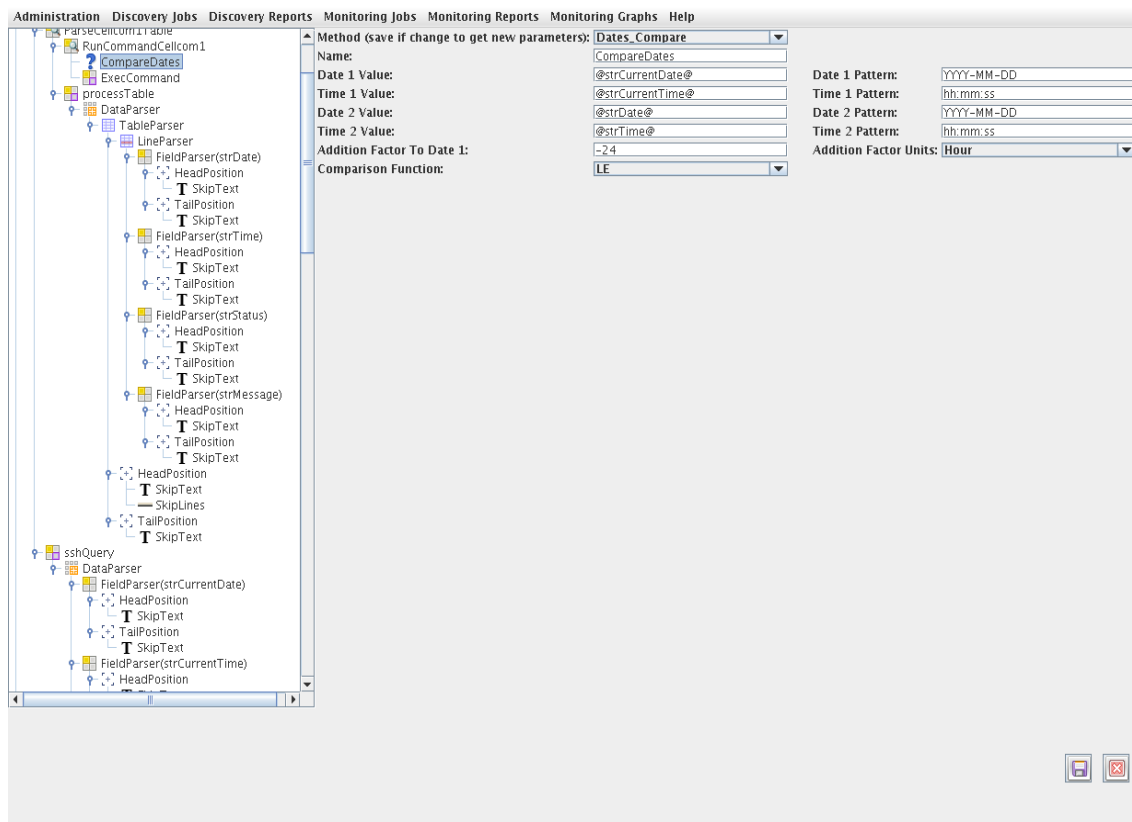
    <strValue>1.3.6.1.4.1.9.1.615</strValue>

    <strFunction>KeyContainValue</strFunction>

</Condition>
```

## Dates Compare Condition

This condition allows comparing between 2 dates values. The dates can contain both date and time information or only one of them.



The keys in this condition are:

- Date 1 value: the first date value we want to compare (usually a parameter)
- Date 1 Pattern: the pattern of Date 1 value.
- Date 2 value: the second date value we want to compare (can be a parameter)
- Date 2 Pattern: the pattern of Date 2 value.
- Time 1 value: the first time value we want to compare (usually a parameter)
- Time 1 Pattern: the pattern of Time 1 value.
- Time 2 value: the second time value we want to compare (can be a parameter)
- Time 2 Pattern: the pattern of Time 2 value.
- The Addition Factor for Date1 and Time 1 : a value that is added to the calculated total value

of date1 and time1

- The Addition Factor units: The units of the addition factor. Currently supported values are:
  - Day
  - Hour
  - Minutes
  - Second
- Comparison function : This function specifies the relation between date1 + the addition factor and date2. The supported function values are:
  - Eq – both date values are same
  - GE – Date 1 is greater or equal than date 2
  - GT – Date 1 is greater than date 2
  - LE – Date 1 is smaller or equal than date 2
  - LT – Date 1 is smaller than date 2

## Date Pattern

The date pattern is a string that specifies the position of the date parts.

The following Date parameters are recognized:

- YYYY – a 4 digit year string
- yy – a 2 digit year string
- MM – a 2 digit month string
- mon – the 3 letter short name of the month (Jan,Feb,...)
- DD – a 2 digit day in the month

## Time Pattern

The time pattern is a string that specifies the position of the time parts.

The following Date parameters are recognized:

- hh – a 2 digit hour string
- mm – a 2 digit minutes string

- ss – a 2 digit seconds string

## The logic of using the Date and Time pattern

The Date comparison condition uses the date and time patterns as pointers to the position of the extracted element in the date/time value.

## Discovery Field Item

The discovery field item, represents a field or a column in a set of lines. It contains all the information required for calculating and storing a value in the field.

There are many types of data extraction methods supported by the product, each has its own specific properties while there many common properties to all field items.

The supported extraction methods are:

- Direct – The value given in a property named “value” is taken as the value of the field. The value can be fixed or a parameter extracted in a higher level in the discovery process.
- AutoGenerateInt – This method is used for generating unique key values in rows. If the entry is of a new entry, then the value returned is a unique Integer Value.
- ICMP – IP addresses will be inserted to a field only if they respond to PING
- SNMP – The data will be extracted using the SNMP protocol, when additional parameters will specify the exact OID to extract.
- CSV – The data will be extracted using an SQL that will look at the CSV file translated as a Table.
- SQL – The data will be extracted from an SQL query working on a database supplied in additional parameters.
- WMI – The data will be extracted using selects from WMI agents located on windows machines (either local or remote).
- Registry – The data will be extracted from the registry of the target machine
- Telnet – Data extracted after executing a script using the telnet protocol is inserted to a field. The data is usually inserted after activating parsing rules.
- SSH - Data extracted after executing a script using the SSH protocol is inserted to a field. The data is usually inserted after activating parsing rules.
- HTTP – Data extracted from a URL is inserted to a field. The data is usually inserted after activating parsing rules.

- Open TCP port – A port number is inserted only if it is possible to connect to the given port number
- LDAP – The data will be extracted using selects from an LDAP repository.

In the following sections we will explain what are the properties needed in order to define each of these field item types.

## Data parsers

On the string results of any field item we can execute a data parsing operation. The data parsing extracts a table of values from the field data. This data can either be used directly or using an SQL query to be inserted later into the requested field value.

We will document in a later section all the functionality of the data parser, and how to use it.

## Common Field Item properties

In this section we will list all the common properties in the field items.

### ***Method***

The Field discovery method defines how the field content is calculated. Currently there are several methods supported, allowing to extract data from multiple data sources.

The Valid values are:

- AutoGenerateInt
- CSV
- Direct
- DiscoverIpRange
- DiscoverSNMPCommunity
- IsTelnetSSHResponding
- LDAP
- OpenTcpPorts
- SNMP

- SystemCommand
- Registry
- SQL
- TelnetSSH
- WMI

### ***Field Type***

This Field specifies what is the type of the field.

Valid Values: String , Int

### ***Table Field Name***

The name of the field in the table in it the value will be stored.

Valid values: Valid field names of the target table in the database

### ***Table Field Description***


The description of the table field, used for documentation.

### ***isKey***

This field specifies if the given field is a part of the current line key.

There could be multiple entries with this key set to 1

Valid values: 0,1

 The first FieldItem entries in the DiscoveryItem, must be the key entries, otherwise the fieldItems defined before will not be entered into the record. This is critical in DiscoveryItems of the type "Table".

### ***isTableKey***

This field specifies if the given field is a part of a key when extracting multiple values in the current

discovery item. This property is relevant for discovery items of the type of 'table'.

For example using SNMP extraction on an interface table the ifIndex is used as a table key in the extraction process, and as such it exists in every field selected so values can be matched to their exact line.

Valid values: 0,1

### ***Unique field check – sql query***

This property is relevant for key fields.

This property contains an SQL query that extracts a value if a record we want to insert/update already exists. The first selected value is the value of the key.

### ***Final SQL query***

When we use a data parser to extract specific data from the retrieved data source, the output in many cases is a table, that can then be queried using the final SQL query.

In this property we insert the actual sql query.

### ***Final SQL selected field name***

This is the property that specifies the field name in the final SQL that represents the value of the field we want to update in the field item.

### ***Final SQL Key field name***

When we are in a 'table' discovery item, and extract multiple values, this property specifies the table key value used to identify fields that are on the same line in the multi line definition.

### ***Skip Protocol If Fails***

This property specifies that if a current extraction failed do not try to continue extracting data from the target data source in other fields. For example if a node does not respond to SNMP for a field that always has a value, then there is no point on continuing polling him in this current discovery cycle.



### ***ID***

This field only appears in the XML of the discovery item, and contains a unique ID between all discovery item elements. It is automatically generated if there none existing.

## Direct Field Item

The direct field item, inserts a specific value specified in the value property into the table's field.

Method (save if change to get new parameters):	Direct
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	strGroupingLevel1
Table field description:	
Unique field check SQL query:	
Value:	@strGroupingLevel1@
Value function:	None
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	





Specific fields:

## ***Value***

The value field contains the value that will be inserted in to the table field.

The value parameter can be a final value (String or Integer) that should fit with the sql valid values or a parameter name (enclosed within 2 @). In this case the parameter name will be replaced with the value as passed by its parent.

## ***Value Function***

The Value Function is a name of a pre-determined function that returns a value that will be put in the specified field.

Currently supported value functions are:

- **CurrentTime** – Returns the current time in an integer format with the number of seconds from 1.1.1970
- **GetHostByAddress** – returns the DNS name of a given host when its IP address is found in a the value property.
- **CalculateSubnetMask** – returns the subnet mask derived from the IP address supplied. The IP address is found in the value property.
- **CalculatSubnetAddr** – returns the subnet address of a given Ip address with a given subnet mask, where the IP address and the subnet mask are found in the value property in a CSV format.
- **GeneratelpRange** – returns a list of Ips based on range value specified in the value property

## ***'Direct' Method XML format***

Following are examples of using the 'Direct' Method in the field Item.

## ***A simple value XML***

```
<FieldItem>

    <strId>68baca9d-6db6-403f-a245-40f8f6ddafae</strId>

    <strMethod>Direct</strMethod>

    <strTableField>strType</strTableField>

    <strTableFieldDescription></strTableFieldDescription>
```

```

    <strFieldType>String</strFieldType>

    <strValue>Node</strValue>

    <strUniqueFieldQuery></strUniqueFieldQuery>

    <strValueFunction>None</strValueFunction>

    <strFinalSQLQuery>null</strFinalSQLQuery>

    <strFinalSQLFieldName>null</strFinalSQLFieldName>

    <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

    <fIsKey>0</fIsKey>

    <fSkipProtocolIfFails>0</fSkipProtocolIfFails>

    <fIsTableKey>0</fIsTableKey>

    <strTableKeyFieldName></strTableKeyFieldName>

</FieldItem>

```

## ***A Time value XML***

```

<FieldItem>

    <strId>32d6a235-c378-41d1-91b1-e383b6b9fd4e</strId>

    <strMethod>Direct</strMethod>

    <strTableField>nUpdateTime</strTableField>

    <strTableFieldDescription></strTableFieldDescription>

    <strFieldType>Int</strFieldType>

    <strValue></strValue>

    <strUniqueFieldQuery></strUniqueFieldQuery>

    <strValueFunction>CurrentTime()</strValueFunction>

    <strFinalSQLQuery>null</strFinalSQLQuery>

    <strFinalSQLFieldName>null</strFinalSQLFieldName>

    <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>



```

```
<fIsKey>0</fIsKey>  
  
<fSkipProtocolIfFails>0</fSkipProtocolIfFails>  
  
<fIsTableKey>0</fIsTableKey>  
  
<strTableKeyFieldName></strTableKeyFieldName>  
  
</FieldItem>
```

## AutoGenerateInt Field Item

The AutoGenerateInt field item, inserts a unique integer value into the target field.

Method (save if change to get new parameters):	AutoGenerateInt
Field Type:	Int
Is field a key:	1
Is field a table key (For Table Discovery Items):	0
Table field name:	nObjectId
Table field description:	
Unique field check SQL query:	<pre>SELECT distinct nParentObjectId FROM tblObject WHERE strType = 'Address' AND stripAddr = '@stripAddr@'</pre>
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



This field item does not have any unique fields.

## **'AutoGenerateInt' Method XML format**

Following is the XML format of the AutoGenerateInt field item.

```
<FieldItem>

  <strId>8f351491-69e2-4b40-b228-0ef2b6b48353</strId>

  <strMethod>AutoGenerateInt</strMethod>

  <strTableField>nObjectId</strTableField>

  <strTableFieldDescription></strTableFieldDescription>

  <strFieldType>Int</strFieldType>

  <strValue></strValue>

  <strUniqueFieldQuery>SELECT
distinct nParentObjectId
FROM tblObject
WHERE
strType = 'Address' AND
strIpAddr = '@strIpAddr@'

  </strUniqueFieldQuery>

  <strValueFunction>None</strValueFunction>

  <strFinalSQLQuery>null</strFinalSQLQuery>

  <strFinalSQLFieldName>null</strFinalSQLFieldName>

  <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

  <fIsKey>1</fIsKey>

  <fSkipProtocolIfFails>0</fSkipProtocolIfFails>

  <fIsTableKey>0</fIsTableKey>

  <strTableKeyFieldName></strTableKeyFieldName>

</FieldItem>
```

## CSV Field item

The CSV field item, extracts values from a CSV file (for example an Excel file). It looks at the CSV file as a table, and allows querying the table. It either assumes that the first line of the table contains the field names or if not then the field names are 'field1', 'field2' , etc.















Method (save if change to get new parameters):	CSV
Field Type:	String
Is field a key:	1
Is field a table key (For Table Discovery Items):	1
Table field name:	strSite
Table field description:	
	select strSite FROM tblSites
Unique field check SQL query:	
URL:	http://127.0.0.1/site_list.txt
separator:	,
User:	
Password:	
	select site from #TEMPTABLE#
SQL Command:	
Query field name:	strSite
Is first line in file, a header line:	1
Key field name in retrieved table (relevant only for table type items):	strSite



Specific fields:

### ***URL***

The url location of the CSV, it can also be in a [file:///](#) format however the file should be located on the machine that runs the discovery process.

### ***User***

The user name value when the http request requires authentication.

### ***Password***

The password value when the http request requires authentication.

### ***SQL command***

The SQL command that will be executed on the CSV content in order to extract the requested values

### ***Query field name***

The name of the field selected in the SQL that represents the value of the field item

### ***Is First line a header***

Indicates if the first line of the CSV file contain field names and if yes they can be used in the SQL query. If not then the field names will be field1,field2,...

### ***Key field name***

When we are in a 'table' discovery item, and extract multiple values, this property specified the table key value used to identify fields that are on the same line in the multi line definition.

### ***'CSV' Method XML format***

Following is the XML format of the CSV field item.

```
<FieldItem>
  <strId>ff840524-fa64-4905-acb1-06c79c6ad41d</strId>
  <strMethod>CSV</strMethod>
  <strTableField>strSite</strTableField>
  <strTableFieldDescription></strTableFieldDescription>
  <strFieldType>String</strFieldType>
  <strValue></strValue>
```

```
<strUniqueFieldQuery>select strSite FROM tblSites</strUniqueFieldQuery>  
  
<strValueFunction></strValueFunction>  
  
<strFinalSQLQuery>null</strFinalSQLQuery>  
  
<strFinalSQLFieldName>null</strFinalSQLFieldName>  
  
<strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>  
  
<flsKey>1</flsKey>  
  
<fSkipProtocolIfFails>0</fSkipProtocolIfFails>  
  
<flsTableKey>1</flsTableKey>  
  
<strTableKeyFieldName>strSite</strTableKeyFieldName>  
  
<strURL>http://127.0.0.1/site_list.txt</strURL>  
  
<strSeperator>,</strSeperator>  
  
<strUser></strUser>  
  
<strPassword></strPassword>  
  
<strCommand>select site from #TEMPTABLE#</strCommand>  
  
<strSQLQueryFieldName>strSite</strSQLQueryFieldName>  
  
<flsFirstLineHeader>1</flsFirstLineHeader>  
  
</FieldItem>
```



## **DiscoverIpRange Field item**

The discover IP range field item, returns IP addresses that respond to ping in given ranges.





Method (save if change to get new parameters):	DiscoverIpRange
Field Type:	String
Is field a key:	1
Is field a table key (For Table Discovery Items):	1
Table field name:	strIpAddr
Table field description:	<div>select strIpAddr from tblNodes</div>
Unique field check SQL query:	
Ip Range List (Range Format: a.b.c* or a.b.c.x-y, List is CSV):	10.0.1.*
Timeout in Millisec	
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

### ***Ip Range list***

This is a CSV of IP ranges, when an IP range value can be in one of the following formats:

- Ip Address – example : 10.0.0.1
- Ip group using masks: 10.1.0.\* takes all values from 10.1.0.0 to 10.1.0.255
- Ip group using ranges: 10.1.0.5-30 takes all values from 10.1.0.5 to 10.1.0.30

### ***Timeout in ms***

This value specified how long to wait for a ping response in milliseconds

### ***'DiscoverIpRange' Method XML format***

Following is the XML format of the DiscoverIpRange field item.

```
<FieldItem>

  <strId>0e261a57-b664-49a5-ba91-72c588f739ee</strId>

  <strMethod>DiscoverIpRange</strMethod>

  <strTableField>strIpAddr</strTableField>

  <strTableFieldDescription></strTableFieldDescription>

  <strFieldType>String</strFieldType>

  <strValue></strValue>

  <strUniqueFieldQuery>select strIpAddr from tblNodes</strUniqueFieldQuery>

  <strValueFunction>None</strValueFunction>

  <strFinalSQLQuery>null</strFinalSQLQuery>

  <strFinalSQLFieldName>null</strFinalSQLFieldName>

  <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

  <flsKey>1</flsKey>

  <fSkipProtocolIfFails>0</fSkipProtocolIfFails>

  <flsTableKey>1</flsTableKey>

  <strTableKeyFieldName></strTableKeyFieldName>
```

```
<strIpRangeList>@IpRangeList@</strIpRangeList>



<nTimeoutMs>1000</nTimeoutMs>

</FieldItem>
```

## Discover SNMP Community Field item

This field item is used to discover a node's SNMP community when taken from a list of options specified in a given property.

Method (save if change to get new parameters):	DiscoverSNMPCommunity
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	strCommunity
Table field description:	
Unique field check SQL query:	
IP Address:	@strIpAddr@
Community List (CSV):	public
SNMP Version (1 2 3):	1
Timeout (msec):	3000
Retry count:	3
Max PDU Size:	64000
Skip Protocol for node on failure(Valid only for Line Items)	0



Specific fields:

### ***Ip Address***

The IP address of the node we want to check the communities on.

### ***Community list***

A CSV of community values.

### ***SNMP Version***

The name of the SNMP version on which this query will work. Currently only version 1 & 2 are supported, however If there is a need for a support of version 3, please contact us.

### ***Timeout in ms***

Timeout for the reply of the SNMP request in milliseconds.

### ***Retry count***

A count that indicates how many times should a request be sent before it is decided that the node does not respond.

### ***'DiscoverSNMPCommunity Method XML format***

Following is the XML format of the DiscoverSNMPCommunity field item.



```
<FieldItem>
  <strId>03258b2b-73f7-4267-9201-386d3f6c5e98</strId>
  <strMethod>DiscoverSNMPCommunity</strMethod>
  <strTableField>strCommunity</strTableField>
  <strTableFieldDescription></strTableFieldDescription>
  <strFieldType>String</strFieldType>
  <strValue></strValue>
  <strUniqueFieldQuery></strUniqueFieldQuery>
  <strValueFunction></strValueFunction>
```

```
<strFinalSQLQuery>null</strFinalSQLQuery>
<strFinalSQLFieldName>null</strFinalSQLFieldName>
<strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>
<flsKey>0</flsKey>
<fSkipProtocolIfFails>0</fSkipProtocolIfFails>
<flsTableKey>0</flsTableKey>
<strTableKeyFieldName></strTableKeyFieldName>
<strIpAddr>@strIpAddr@</strIpAddr>
<strCommunityList>public</strCommunityList>
<nSNMPVersion>1</nSNMPVersion>
<nTimeoutMsec>3000</nTimeoutMsec>
<nRetries>3</nRetries>
<nMaxPDUSize>64000</nMaxPDUSize>
</FieldItem>
```

## Is Telnet SSH Responding Field item

This field item is used to discover if the connection information passed for Telnet or SSH protocols is valid and if the target node is responding.

Method (save if change to get new parameters):	IsTelnetSSHResponding
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	IsTelnetSSHResponding
Table field description:	
Unique field check SQL query:	
Protocol:	SSH2
Host name or IP:	@stripAddr@
Port:	
User:	user
Password:	password
SSH Autentication method:	Password
SSH PEM file:	



Specific fields:

### **Protocol**

The 2 protocols supported are Telnet & SSH.

***Host name or IP***

The address of DNS name of the node to test.

***Port***

The port to connect to, the default for ssh is 22 and for telnet is 21.

***User***

The user name used for authentication\

***Password***

The password used for authentication

***SSH authentication method***

There are 2 methods of authentication supported for SSH:

- Password – requires the user & password fields
- PKI – requires a manual generation of a key for the station running the discovery, and the allowing of that station in the target.

***SSH PEM file***

The authentication information file used when working is PKI authentication.

***'IsTelnetSSHResponding Method XML format***

Following is the XML format of the DiscoverSNMPCommunity field item.

```
<FieldItem>
  <strId>7de0ef38-e420-45e7-aa6b-321ee645f4b0</strId>
  <strMethod>IsTelnetSSHResponding</strMethod>
  <strTableField>IsTelnetSSHResponding</strTableField>
  <strTableFieldDescription></strTableFieldDescription>
  <strFieldType>String</strFieldType>
  <strValue></strValue>
  <strUniqueFieldQuery></strUniqueFieldQuery>
```

```
<strValueFunction></strValueFunction>

<strFinalSQLQuery>null</strFinalSQLQuery>

<strFinalSQLFieldName>null</strFinalSQLFieldName>

<strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

<flsKey>0</flsKey>

<fSkipProtocolIfFails>0</fSkipProtocolIfFails>

<flsTableKey>0</flsTableKey>

<strTableKeyFieldName></strTableKeyFieldName>

<strHostName>@strIpAddr@</strHostName>

<nPort></nPort>

<strScriptName></strScriptName>

<strScriptData></strScriptData>

<strPrompt>null</strPrompt>

<strProtocol>SSH2</strProtocol>

<strUser>user</strUser>

<strPassword>password</strPassword>

<strAuthMethod>Password</strAuthMethod>

<strScriptParameters>null</strScriptParameters>

<strPemFile>null</strPemFile>



</FieldItem>
```



## LDAP Field item

This field item is used to discover data located in LDAP repositories.

Method (save if change to get new parameters):	LDAP
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	LDAPField
Table field description:	
Unique field check SQL query:	
Ldap URL:	ldap://ldap.example.com:389
User:	user
Password:	password
LDAP Source:	DC = jilroy,DC = com
LDAP query:	(objectclass=*)
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

Specific fields:

### ***Ldap URL***

The URL of the LDAP repository with a possible port addition.

**User**

The user name used for authentication with the Ldap repository

**Password**

The password used for the authentication with the Ldap repository.

**LDAP Source**

The Source from which the LDAP query will be executed.

**LDAP Query**

The LDAP query command that will be executed.

**LDAP Method XML format**

Following is the XML format of the LDAP field item.



```
<FieldItem>
  <strId>2d81a4dd-b0cb-4268-8deb-52564515be08</strId>
  <strMethod>LDAP</strMethod>
  <strTableField>LDAPField</strTableField>
  <strTableFieldDescription></strTableFieldDescription>
  <strFieldType>String</strFieldType>
  <strValue></strValue>
  <strUniqueFieldQuery></strUniqueFieldQuery>
  <strValueFunction></strValueFunction>
  <strFinalSQLQuery>null</strFinalSQLQuery>
  <strFinalSQLFieldName>null</strFinalSQLFieldName>
  <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>
  <flsKey>0</flsKey>
```

```
<fSkipProtocolIfFails>0</fSkipProtocolIfFails>  
<flsTableKey>0</flsTableKey>  
<strTableKeyFieldName></strTableKeyFieldName>  
<strUser>user</strUser>  
<strPassword>password</strPassword>  
<strCommand>(objectclass=*)</strCommand>  
<strSource>DC=jilroy,DC=com</strSource>  
<strURL>ldap://ldap.example.com:389</strURL>  
</FieldItem>
```

## Open Tcp Ports Field item

This field item is used to discover open ports within a given node. It returns the open ports within the supplied port range.

Method (save if change to get new parameters):	OpenTcpPorts
Field Type:	String
Is field a key:	1
Is field a table key (For Table Discovery Items):	1
Table field name:	nOpenPort
Table field description:	
Unique field check SQL query:	
Ip Address:	@stripAddr@
Port Range List (Range Format: start-end,port, List is CSV):	1-1000
Timeout in Millisec	
Max concurrent socket checks:	
Final SQL Query:	<pre>select stripAddr, nPort from #TEMPTABLE#</pre>
Final SQL Selected field name:	nPort
Final SQL Key field name in retrieved table:	nPort



Specific fields:

### **IP Address**

The IP address on which to look for open ports.

## ***Port Range list***

A CSV of port ranges, where a range is in a format of x-y where x & y are valid port numbers and x < y or in a format of a number if only one port should be checked.

## ***Timeout in milliseconds***

The timeout we want to wait for a connection to be established until we fail. The default is 20 sec (20000).

## ***Max Concurrent socket checks***

This parameter specifies how many ports we will check concurrently. This parameter determines the speed of the check but depends on the machine capability. The default value is 1000.

## **Open TCP Ports Method XML format**

Following is the XML format of the Open TCP ports field item.

```
<FieldItem>

    <strId>8b9451a4-ec10-4559-a72e-76e1807b3170</strId>

    <strMethod>OpenTcpPorts</strMethod>

    <strTableField>nOpenPort</strTableField>

    <strTableFieldDescription></strTableFieldDescription>

    <strFieldType>String</strFieldType>

    <strValue></strValue>

    <strUniqueFieldQuery></strUniqueFieldQuery>

    <strValueFunction>None</strValueFunction>

    <strFinalSQLQuery>select strIpAddr, nPort from #TEMPTABLE#</strFinalSQLQuery>

    <strFinalSQLFieldName>nPort</strFinalSQLFieldName>

    <strFinalSQLTableKeyName>nPort</strFinalSQLTableKeyName>

    <flsKey>1</flsKey>

    <fSkipProtocolIfFails>0</fSkipProtocolIfFails>

    <flsTableKey>1</flsTableKey>



    <strTableKeyFieldName></strTableKeyFieldName>
```

```
<stripAddr>@stripAddr@</stripAddr>  
  
<strPortRangeList>1-1000</strPortRangeList>  
  
<nMaxConcurrentSockets>1000</nMaxConcurrentSockets>  
  
<nTimeoutMs>10000</nTimeoutMs>  
  
</FieldItem>
```

## SNMP Field item

This field item returns values located in the SNMP mib of the destination node.

Method (save if change to get new parameters):	SNMP
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	strSysObjectId
Table field description:	
Unique field check SQL query:	
IP Address:	@strIpAddr@
Community:	@strCommunity@
SNMP Version (1 2 3):	1
Timeout (msec):	3000
Retry count:	
Max PDU Size:	64000
OID:	1.3.6.1.2.1.1.2.0
OID description:	mib2.system.sysObjectId
Can data be in Hex format:	0
Get key value of snmp reply:	0
Skip Protocol for node on failure(Valid only for Line Items)	1
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

### **IP Address**

The IP address to SNMP query

***Community***

The SNMP community to use

***SNMP Version***

The SNMP version to use. Currently only version 1 & 2 are supported

***Timeout in milliseconds***

The timeout value, that will be used to wait for an SNMP reply

***Retry count***

The retry count when timeout occurs and no reply received.

***Max PDU size***

The maximum PDU size received. Default is 64K.

***OID***

The OID requested. This value can be a prefix of the OID too, and an SNMP walk will be made to the given tree.

***OID description***

A descriptive field that describes the OID requested.

***Can data be in hex format***

Some times the value returned is in Hex format. If this field is set to true, then the value is converted to the string representation of the hex value.

***Get Key value of SNMP reply***

Some times the value requested is the specific key returned and not the value of the field. This is the case in get table operations. When this property is set to true, then the key value is returned.

***Skip protocol for node in case of an Error***

If an error occurs using this protocol, returning a field that must always exist, then specifying true in this field, will prevent querying the node using this protocol in other fields of this discovery cycle.

It makes sense to lit that flag on a field that always exists in the mib, and always has a value.



## ***SNMP Method XML format***

Following is the XML format of the SNMP field item.

```
<FieldItem>

  <strId>55cc9283-91bb-4e5a-9682-81eafa4a60a9</strId>

  <strMethod>SNMP</strMethod>

  <strTableField>strSysObjectId</strTableField>

  <strTableFieldDescription></strTableFieldDescription>

  <strFieldType>String</strFieldType>

  <strValue></strValue>

  <strUniqueFieldQuery></strUniqueFieldQuery>

  <strValueFunction>None</strValueFunction>

  <strFinalSQLQuery>null</strFinalSQLQuery>

  <strFinalSQLFieldName>null</strFinalSQLFieldName>

  <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

  <fIsKey>0</fIsKey>

  <fSkipProtocolIfFails>1</fSkipProtocolIfFails>

  <fIsTableKey>0</fIsTableKey>

  <strTableKeyFieldName></strTableKeyFieldName>

  <strIpAddr>@strIpAddr@</strIpAddr>

  <strCommunity>@strCommunity@</strCommunity>

  <nSNMPVersion>1</nSNMPVersion>

  <nTimeoutMsec>3000</nTimeoutMsec>

  <nRetries>3</nRetries>

  <nMaxPDUSize>64000</nMaxPDUSize>



  <strOID>1.3.6.1.2.1.1.2.0</strOID>
```

```
<strOIDName>mib2.system.sysObjectId</strOIDName>  
  
<fIsPossibleHexData>0</fIsPossibleHexData>  
  
<fGetKeyValue>0</fGetKeyValue>  
  
</FieldItem>
```

## Registry Field item

This field item returns values located in the Registry repository in the target node.

Method (save if change to get new parameters):	Registry
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	RegistryField
Table field description:	
Unique field check SQL query:	
Root Key:	HK_ROOT
WMI Path:	/
Query Type:	Key
User:	user
Password:	password
Ip Address:	@stripAddr@
Requested field name:	
Key field name in retrieved table (relevant only for table type items):	
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

### Root Key

The registry root key. For example: HK\_ROOT, HK\_CURRENT\_USER, etc.

***WMI Path***

The path of the requested value in the current registry root.

***Query type***

The type of the query. It can be for a directory content (Key) or for a value of a given key (Field)

***User***

The user name used for authentication.

***Password***

The password value used for authentication.

***IP***

The address of the queried node. A local node must be marked as 127.0.0.1

***Field name***

When multiple values are retrieved, then the field requested should be specified in this property.

***Key field name***

When we are in a 'table' discovery item, and extract multiple values, this property specifies the table key value used to identify fields that are on the same line in the multi line definition.

## ***Registry Method XML format***

Following is the XML format of the Registry field item.

```
<FieldItem>

    <strId>b974c212-2f6c-44e0-a958-e7e3a3f656aa</strId>

    <strMethod>Registry</strMethod>

    <strTableField>RegistryField</strTableField>

    <strTableFieldDescription></strTableFieldDescription>

    <strFieldType>String</strFieldType>

    <strValue></strValue>

    <strUniqueFieldQuery></strUniqueFieldQuery>

    <strValueFunction></strValueFunction>

    <strFinalSQLQuery>null</strFinalSQLQuery>

    <strFinalSQLFieldName>null</strFinalSQLFieldName>

    <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

    <flsKey>0</flsKey>

    <fSkipProtocolIfFails>0</fSkipProtocolIfFails>

    <flsTableKey>0</flsTableKey>

    <strTableKeyFieldName></strTableKeyFieldName>

    <strIpAddr>@strIpAddr@</strIpAddr>

    <strUser>user</strUser>

    <strPassword>password</strPassword>

    <strRootKey>HK_ROOT</strRootKey>

    <strPath></strPath>

    <strQueryType>Key</strQueryType>



    <strRegistryFieldName></strRegistryFieldName>

</FieldItem>
```

## SQL Field item

This field item returns values located in SQL repositories.

Method (save if change to get new parameters):	SQL
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	SQLField
Table field description:	
Unique field check SQL query:	
SQL Driver Name:	org.apache.derby.jdbc.ClientDriver
Connection String:	jdbc:derby://127.0.0.1:54345/rjnetworkdiscov
User:	user
Password:	password
SQL Command:	select strSysObjectId from tblObject
Query field name:	strSysObjectId
Key field name in retrieved table (relevant only for table type items):	



Specific fields:

### **SQL Driver name**

The JDBC driver used for connecting to the database

***SQL Connection string***

The connection string to the database.

***User***

The user name used to authenticate the database

***Password***

The password value used to authenticate the database

database

***SQL command***

The SQL command used to retrieve the data. Multiple fields may be returned. Those fields can be used in later child discovery items.

***Query field name***

The name of the field extracted in the SQL command that contains the value needed to fill our field item.

***Table Key field***

When we are in a 'table' discovery item, and extract multiple values, this property specified the table key value used to identify fields that are on the same line in the multi line definition.

***SQL Method XML format***

Following is the XML format of the SQL field item.

```
<FieldItem>
  <strId>4d06dccc-8585-413f-9d39-8b32a2f64e59</strId>
  <strMethod>SQL</strMethod>
  <strTableField>SQLField</strTableField>
  <strTableFieldDescription></strTableFieldDescription>
  <strFieldType>String</strFieldType>
  <strValue></strValue>
  <strUniqueFieldQuery></strUniqueFieldQuery>
  <strValueFunction></strValueFunction>
  <strFinalSQLQuery>null</strFinalSQLQuery>
```

```

<strFinalSQLFieldName>null</strFinalSQLFieldName>

<strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

<flsKey>0</flsKey>

<fSkipProtocolIfFails>0</fSkipProtocolIfFails>

<flsTableKey>0</flsTableKey>

<strTableKeyFieldName></strTableKeyFieldName>

<strDriverName>org.apache.derby.jdbc.ClientDriver</strDriverName>

<strManager>jdbc:derby://127.0.0.1:54345/rjnetworkdiscovery</strManager>

<strUser>user</strUser>

<strPassword>password</strPassword>

<strCommand>select strSysObjectId from tblObject</strCommand>

<strSQLQueryFieldName>strSysObjectId</strSQLQueryFieldName>Timeout in ms

```

Timeout for the reply of the SNMP request in milliseconds.

### ***Retry count***

A count that indicates how many times should a request be sent before it is decided that the node does not respond.

```
</FieldItem>
```





## Telnet / SSH Field item

This field item returns values using the telnet or the SSH protocol.

This field item must use the data parser tools in order to extract values from the returned data.

Method (save if change to get new parameters):	TelnetSSH
Field Type:	String
Is field a key:	1
Is field a table key (For Table Discovery Items):	1
Table field name:	strFileName
Table field description:	
Unique field check SQL query:	<pre>select strIpAddr, strFileName from tblNodeFile</pre>
Protocol:	SSH2
Host name or IP:	@strIpAddr@
Port:	
User:	user
Password:	password
SSH Authentication method:	Password
SSH PEM file:	
Script file name:	listfiles.txt
Script Data:	
Script parameters:	<pre>path=/home/raphy/ filepath=</pre>
Final SQL Query:	<pre>select strFileName from #TEMPTABLE#</pre>
Final SQL Selected field name:	strFileName
Final SQL Key field name in retrieved table:	strFileName

Specific fields:

**Protocol**

The protocol used in order to connect to target. Currently supported are SSH and Telnet.

**Host name or IP**

The address of DNS name of the node to test.

**Port**

The port to connect to, the default for ssh is 22 and for telnet is 21.

**User**

The user name used for authentication\

**Password**

The password used for authentication

**SSH authentication method**

There are 2 methods of authentication supported for SSH:

- Password – requires the user & password fields
- PKI – requires a manual generation of a key for the station running the discovery, and the allowing of that station in the target.

**SSH PEM file**

The authentication information file used when working is PKI authentication.

**Script file name**

A name of a script found in [INSTALLDIR]/conf/scripts that will be run using the given protocol.

For example the script file may contain the following:

```
cd @path@  
ls -l @filepath@
```

**Script data**

The script data that will be run using the given protocol. This parameter is taken if no value was specified in the 'Script file name' parameter.

This property is used for simple scripts.

**Script parameters**

The data in the script can be changed prior to its run, and parameters specified in it will be replaced by

values given in this property.

### ***Control commands within the script data***

You can insert control commands into the script data that control the following:

- Command Prompt
- Command timeout
- Fail on Timeout state

You insert the commands as displayed in the following example

```
#Prompt=$  
#RegexPrompt=w+$  
#CommandTimeout=60  
#FailOnError=1  
telnet/ssh command
```

You can insert these commands multiple times within the commands stream, and they remain until they are overridden by a new instance

### **Prompt**

Prompt is a string that should appear within the data as is. It is searched from the last command output.

### **RegexPrompt**

RegexPrompt is a regular expression that is searched in the data from the last command output

### **CommandTimeout**

A numerical value (in seconds) that defines an upper limit to the time we wait for the requested prompt. The search will end when the prompt is found

### **FailOnError**

If the timeout for the command expired, and if FailOnError = 1 then we will abort the script in this condition. A records will be written to the log.

### ***Telnet vs. SSH handling of the script control commands***

There is a difference between the way that the prompt is searched in telnet and in ssh. In telnet the prompt is searched before sending the actual command, and in ssh this is inverse

The reason for that, is that in telnet we might have to search for a prompt before sending the login user-Id information, and as such we need to supply the prompt for this before typing the first data. In SSH there is no need for that.

### ***'TelnetSSH' Method XML format***

Following is the XML format of the TelnetSSH field item.

```

<FieldItem>

    <strId>c4849c40-6e67-4683-a6a3-0c1b9a617189</strId>

    <strMethod>TelnetSSH</strMethod>

    <strTableField>strFileName</strTableField>

    <strTableFieldDescription></strTableFieldDescription>

    <strFieldType>String</strFieldType>

    <strValue></strValue>

    <strUniqueFieldQuery>select strIpAddr, strFileName
from tblNodeFile</strUniqueFieldQuery>

    <strValueFunction></strValueFunction>

    <strFinalSQLQuery>select strFileName
from #TEMPTABLE#</strFinalSQLQuery>

    <strFinalSQLFieldName>strFileName</strFinalSQLFieldName>

    <strFinalSQLTableKeyName>strFileName</strFinalSQLTableKeyName>

    <flsKey>1</flsKey>

    <fSkipProtocolIfFails>0</fSkipProtocolIfFails>

    <flsTableKey>1</flsTableKey>

    <strTableKeyFieldName></strTableKeyFieldName>

    <strHostName>@strIpAddr@</strHostName>

    <nPort></nPort>

    <strScriptName>listfiles.txt</strScriptName>

    <strScriptData></strScriptData>



```

```
<strPrompt>null</strPrompt>
<strProtocol>SSH2</strProtocol>
<strUser>user</strUser>
<strPassword>password</strPassword>
<strAuthMethod>Password</strAuthMethod>
<strScriptParameters>path=/home/raphy/
filepath=
</strScriptParameters>
    <strPemFile>null</strPemFile>
</FieldItem>
```

## System Command Field item

This field item is the one responsible for the “Active” Inventory. It is not used to insert data to the inventory, but is used to perform actions on the data extracted as it is being extracted.

Method (save if change to get new parameters):	SystemCommand
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	ExecCommand
Table field description:	
Unique field check SQL query:	
Script file name:	command\sh
Script Data:	
Script parameters:	strDate=@strDate@ strTime=@strTime@ strStatus=@strStatus@ strMessage=@strMessage@
Single Command Timeout(ms):	
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

### ***Script file name***

The name of a script located in [installdir]/conf/scripts/telnetssh/ directory.

Note: you can specify a script name or the actual commands in the field named 'script data'.

### ***Script data***

This field may contain the actual commands that will be executed as a script.

```
ls -l @filepath@
```

This property is used for simple scripts.

You may either specify data in this field or in the 'script file name' file

### ***Script Parameters***

These parameters will replace @ParmName@ variables in the actual script commands prior to the execution. This enables writing dynamic scripts based on the discovered data.

### ***'SystemCommand Method XML format***

Following is the XML format of the SystemCommand field item.

```
<FieldItem>
  <strId>ec309cc1-4eea-44a2-9e8d-63f18074b2ab</strId>
  <strMethod>SystemCommand</strMethod>
  <strTableField>ExecCommand</strTableField>
  <strTableFieldDescription></strTableFieldDescription>
  <strFieldType>String</strFieldType>
  <strValue></strValue>
  <strUniqueFieldQuery></strUniqueFieldQuery>
  <strValueFunction></strValueFunction>
  <strFinalSQLQuery>null</strFinalSQLQuery>
  <strFinalSQLFieldName>null</strFinalSQLFieldName>
  <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>
```

```
<flsKey>0</flsKey>
<fSkipProtocolIfFails>0</fSkipProtocolIfFails>
<flsTableKey>0</flsTableKey>
<strTableKeyFieldName></strTableKeyFieldName>
<strScriptName>cellcom1.sh</strScriptName>
<strScriptData></strScriptData>
<nSingleCommandTimeoutMs></nSingleCommandTimeoutMs>
<strScriptParameters>strDate=@strDate@
```

```
strTime=@strTime@
```

```
strStatus=@strStatus@
```

```
strMessage=@strMessage@</strScriptParameters>
```

```
</FieldItem>
```



## **WMI Field item**

This field item returns values using the WMI protocol from windows stations.



















Method (save if change to get new parameters):	WMI
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	WMIField
Table field description:	
Unique field check SQL query:	
Host IP Address:	@stripAddr@
WMI Path:	\root\cimv2
User:	user
Password:	password
WMI query:	Select UserName From Win32_ComputerSystem
Query field name:	UserName
WMI Key field name in retrieved table (relevant only for table type items):	UserName
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

### ***Ip Address***

The IP address of the machine to be queried.

### ***Path***

The path to the WMI repository

**User**

The user name used for the remote WMI authentication

**Password**

The password value used for the remote WMI authentication

**Query command**

The WMI select command used to select data from the WMI repository

**Field Name**

The name of the field in the result set that will be taken as the value of the field.

**Table Key Field**

When we are in a 'table' discovery item, and extract multiple values, this property specified the table key value used to identify fields that are on the same line in the multi line definition.

**'WMI' Method XML format**

Following is the XML format of the WMI field item.

```
<FieldItem>

  <strId>54a7d78e-b915-43bf-aead-0b6d61b9ce75</strId>

  <strMethod>WMI</strMethod>

  <strTableField>WMIField</strTableField>

  <strTableFieldDescription></strTableFieldDescription>

  <strFieldType>String</strFieldType>

  <strValue></strValue>

  <strUniqueFieldQuery></strUniqueFieldQuery>

  <strValueFunction></strValueFunction>

  <strFinalSQLQuery>null</strFinalSQLQuery>

  <strFinalSQLFieldName>null</strFinalSQLFieldName>

  <strFinalSQLTableKeyName>null</strFinalSQLTableKeyName>

  <flsKey>0</flsKey>

  <fSkipProtocolIfFails>0</fSkipProtocolIfFails>
```



```
<flsTableKey>0</flsTableKey>  
  
<strTableKeyFieldName>UserName</strTableKeyFieldName>  
  
<stripAddr>@stripAddr@</stripAddr>  
  
<strUser>user</strUser>  
  
<strPassword>password</strPassword>  
  
<strCommand>Select UserName From Win32_ComputerSystem</strCommand>  
  
<strPath>\root\cimv2</strPath>  
  
<strWMIQueryFieldName>UserName</strWMIQueryFieldName>  
  
</FieldItem>
```

## Async Ping Field item

This field item returns is used for high speed discovery processes in very large scale organizations. It is commonly used to find if a given list of IP addresses are responding to ICMP echo requests (ping).

The IP list is found under a given field name, in the currently built table. The value returned is an integer when (0 – non responding, 1 – responding). The algorithm this field item implements is extremely fast, and it can maps 100,000s of addresses within a few minutes based on specified flow control parameters passed to the command.

Method (save if change to get new parameters):	AsyncPing
Field Type:	Int
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	nlIsResponding
Table field description:	
Unique field check SQL query:	
Ip List Field Name (in current line or table):	stripAddr
Timeout in Millisec	3000
Retry count:	3
Window size count:	5000
requests count sent in a second:	100
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

***Ip List Field Name***

The IP Address field name in the currently defined discovery item. The IP field item must be defined before this field item.

***Timeout in ms***

Timeout for the reply of the ICMP request in milliseconds.

***Retry count***

A count that indicates how many times should a request be sent before it is decided that the node does not respond.

***Window Size***

This is a flow control parameter. A count of how many ICMP requests can be open at a single time.

***Requests count sent in one second***



This is a flow control parameter. This count determines how many requests will be sent within one second.

## Async Discover SNMP Community Field item

This field item returns is used for high speed discovery processes in very large scale organizations. It is commonly used to find if a list of IPs responds to SNMP and what is the community they respond to.

The IP list is found under a given field name, in the currently built table. The value returned is the responding community string that the IP responds to. The algorithm this field item implements is extremely fast, and it can maps 100,000s of addresses within a few minutes based on specified flow control parameters passed to the command.

Method (save if change to get new parameters):	AsyncDiscoverSNMPCommunity
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	strCommunity
Table field description:	
Unique field check SQL query:	
Ip List Field Name (in current line or table):	stripAddr
Community List (CSV):	public,private
SNMP Version (1 2 3):	1
Timeout (msec):	3000
Retry count:	3
Max PDU Size:	64000
requests count sent in a second:	100



Specific fields:

***Ip List Field Name***

The IP Address field name in the currently defined discovery item. The IP field item must be defined before this field item.

***Community list***

The community list to check in a CSV (comma separated vector) format.

***Timeout in ms***

Timeout for the reply of the SNMP request in milliseconds.

***Retry count***

A count that indicates how many times should a request be sent before it is decided that the node does not respond.

***Window Size***

This is a flow control parameter. A count of how many SNMP requests can be open at a single time.

***Requests count sent in one second***



This is a flow control parameter. This count determines how many requests will be sent within one second.

## Async SNMP Field item

This field item returns is used for high speed discovery processes in very large scale organizations. It is commonly used to return specific SNMP field values of a given IP list.

The IP list is found under a given field name, in the currently built table. The value returned is the actual value returned from the node.. The algorithm this field item implements is extremely fast, and it can maps 100,000s of addresses within a few minutes based on specified flow control parameters passed to the command.

Method (save if change to get new parameters):	AsyncSNMP
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	strSysObjectId
Table field description:	
Unique field check SQL query:	
Ip List Field Name (in current line or table):	strIpAddr
Community List Field Name:	strCommunity
SNMP Version (1 2 3):	1
Timeout (msec):	3000
Retry count:	3
Max PDU Size:	64000
requests count sent in a second:	100
OID:	1.3.6.1.2.1.1.2.0
OID description:	mib2.system.sysObjectId
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:



***Ip List Field Name***

The IP Address field name in the currently defined discovery item. The IP field item must be defined before this field item.

***Community Field Name***

The community field name that should be used in the SNMP request.

***Timeout in ms***

Timeout for the reply of the SNMP request in milliseconds.

***Retry count***

A count that indicates how many times should a request be sent before it is decided that the node does not respond.

***Window Size***

This is a flow control parameter. A count of how many SNMP requests can be open at a single time.

***Requests count sent in one second***



This is a flow control parameter. This count determines how many requests will be sent within one second.

## Async Open Ports Field item

This field item returns is used for high speed discovery processes in very large scale organizations. It is commonly used to return an indication if a given port is open for TCP connections on a list of IPs.

The IP list is found under a given field name, in the currently built table. The value returned is an integer when (0 – non responding, 1 – responding). The algorithm this field item implements is extremely fast, and it can maps 100,000s of addresses within a few minutes based on specified flow control parameters passed to the command.

Method (save if change to get new parameters):	AsyncOpenTcpPorts
Field Type:	String
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	flsSSH
Table field description:	
Unique field check SQL query:	
Ip List Field Name (in current line or table):	strIpAddr
Port number to check:	22
Timeout in Millisec	10000
Max concurrent socket checks:	1000
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	



Specific fields:

***Ip List Field Name***

The IP Address field name in the currently defined discovery item. The IP field item must be defined before this field item.

***Port Number***

The port number to check

***Timeout in ms***

Timeout for the reply of the SNMP request in milliseconds.

***Max concurrent socket checks***

This is a flow control parameter. A count of how many connect requests can be open.

# The Data parser

---

The data parser is used to parse the data returned by the field item, prior to inserting it into the designated field.

In some cases the data parser is compulsory as in the case of the telnet/ssh field item, and in other cases this is only an option.

It works in conjunction with the final SQL command properties, when the data parser is used to generate the temporary SQL command that these properties work on.



The data parsing is composed of the following objects:

- The data parser object – this is the container that holds the parsing instructions for the current field.
- The data parser element – this object represents the type of data we want to parse. There are 4 supported parser elements:
  - A paragraph
  - A Table

- A Line
- A Field
- Under each parser elements there could be Position elements that mark its start and its end
- Each position element may contain one or more locaters which point to the start/end of the requested element. There are several types of locaters:
  - Text locater
  - Line locater
  - word locater
  - Offset locater

There could be a hierarchic structure of parser elements. For example a paragraph may contain a table data parser. A line can be located within a table parser.

## The Data parser entry

This entry is used as a container of all elements related to a single parse action. It has no properties panel.

It can contain under it only element parsers of the type:

- Paragraph
- Table
- Line
- Field

## The Paragraph parser entry

This entry is used to define a paragraph with in the parsed data.

It can contain:

- Data parser elements of the types:
  - Table
  - Line

- Field
- Position elements of the type:
  - Head position element
  - Tail position element

Its properties panel contain only one value which indicates whether the parsing will continue until the end of the data or not.

## The Table parser entry

This entry is used to define a table with in the parsed data.

It can contain:

- Data parser elements of the types:
  - Line
  - Field
- Position elements of the type:
  - Head position element
  - Tail position element

Its properties panel contain only one value which indicates whether the parsing will continue until the end of the data or not.

## The Line parser entry

This entry is used to define a line with in the parsed data.

It can contain:

- Data parser elements of the type:
  - Field
- Position elements of the type:

- Head position element
- Tail position element

Its properties panel contain only one value which indicates whether the parsing will continue until the end of the data or not.

## The Field parser entry

This entry is used to define a field with in the parsed data.

It can contain:

- A field name
- Position elements of the type:
  - Head position element
  - Tail position element

Its properties panel contain only one value which indicates the name of the field extracted. This value will be used in the final SQL query as the fields name in the temporary table.

## The Regular expression Field Array parser entry

This entry is used to define a set of fields extracted from the available data using a regular expression. The regular expression should be defined in a way that extracts fields out of the data. It can extract any number of fields within the available data, located between the head and tail position.

It can contain:

- A prefix of the field names generated. An Index starting from 0 is enumerating the fields selected.
- Position elements of the type:
  - Head position element
  - Tail position element

Its properties panel contain the following values:

- The prefix of the field names extracted. An index starting with 0 is added to this prefix.

- A regular expression string.
- The maximum number of fields that need to be extracted. This is important because when we define such a fields extractor within a table, the table should have the same number of fields within every line to allow SQL operations on the extracted data.

## The Head Position entry

This entry is used to define a the start of the parser element which it is contained in.

It has no properties however it can contain any number of locaters.

## The Tail Position entry

This entry is used to define a the start of the parser element which it is contained in.

It has no properties however it can contain any number of locaters.

## The locaters

The locaters help locating the position of the requested field.



There are currently 4 types of locaters:

### The text locator

The text locator searches the data in the element



Action when not found:	Ignore
Start search From:	Top
Take instance number:	1
	Date:
The data to look for	
Type of search:	Text



## Start Search from

Indicates from where to start the search

## Take Instance number

Indicates which instance of the data to take.

## The data to look for

The data we search

## Type of search

Is search for regex or for actual string

## Action when not found

What should the program do if data not found. The options are:

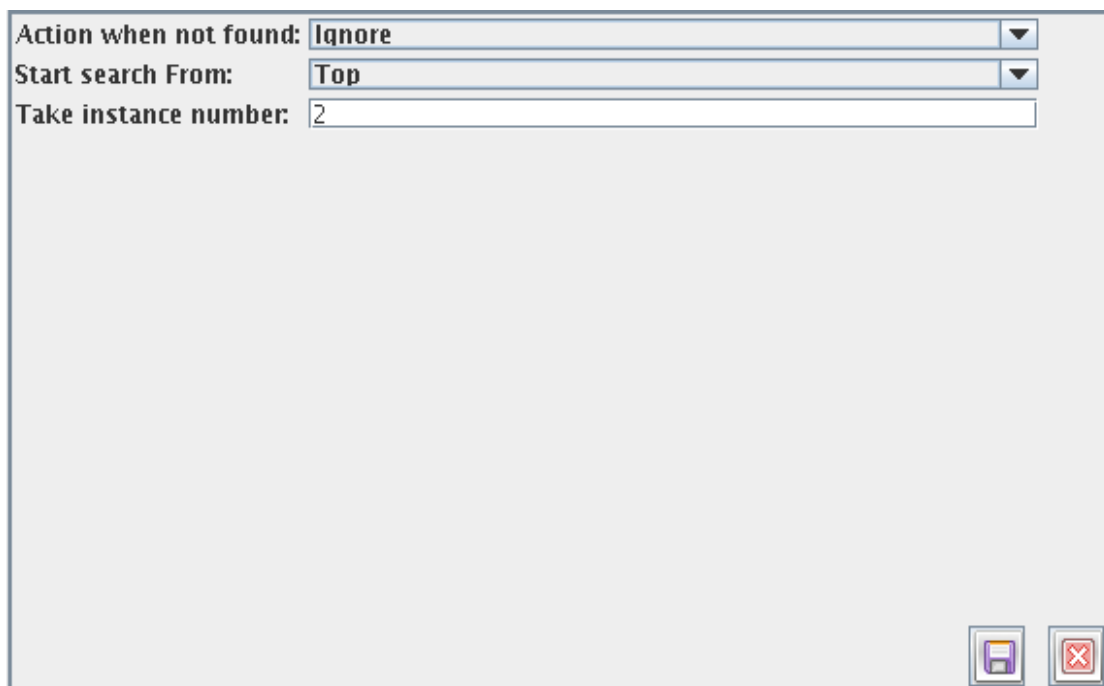
Ignore – do nothing

Log – logs in the file log and in the database table log

Fail – fails the process and log

## The line locator

The line locator searches for a given line of data in the data parsed.



Action when not found: Ignore

Start search From: Top

Take instance number: 2

Its properties are:

### **Start Search from**

Indicates from where to start the search

### **Take Instance number**

Indicates which instance of the data to take.

## Action when not found

What should the program do if data not found. The options are:

Ignore – do nothing

Log – logs in the file log and in the database table log

Fail – fails the process and log

## The word locator

The word locator searches for a given word of data in the data parsed.

Its properties are:

### Start Search from

Indicates from where to start the search

### Take Instance number

Indicates which word number to take

## Action when not found

What should the program do if data not found. The options are:

Ignore – do nothing

Log – logs in the file log and in the database table log

Fail – fails the process and log

## The offset locator

The offset locator searches for a given field based on character offset in the data parsed.

Its properties are:

### Start Search from

Indicates from where to start the search

## **Take Instance number**

Indicates number of letters to skip until the start/end of the field

## **Action when not found**

What should the program do if data not found. The options are:

Ignore – do nothing

Log – logs in the file log and in the database table log

Fail – fails the process and log

# Final Page

---

## More Information

More information about Jilroy Software, Discovery Genie 3.1, and our other products can be found on our web site: [www.jilroy.com](http://www.jilroy.com)

## Contact Us

For any information or problem, request for information or extension idea related to Discovery Genie 3.1, please contact one of the following email addresses.

### Sales

[sales@jilroy.com](mailto:sales@jilroy.com)

### Product Management

[discovery\\_pm@jilroy.com](mailto:discovery_pm@jilroy.com)

### FTP Site

All Jilroy products can be downloaded from our Web site.