

**Discovery Rules**  
*Sample*



**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 © 2000-2012 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>5</b>
<b>Introduction</b> .....	<b>7</b>
Assumptions.....	7
The discovered environment.....	7
What do we want to discover.....	8
The output of the discovery.....	8
tblObject.....	8
tblArp.....	8
tblVlanInfo.....	8
How to view the data.....	9
<b>Explanation of the Sample</b> .....	<b>10</b>
A short reminder about defining discovery rules.....	10
A general description of the discovery items within the example.....	11
ATT_csv_based_discovery.....	13
stripAddr.....	13
SNMPNode.....	15
nObjectId.....	16
strType.....	18
strSysObjectId.....	19
nUpdateTime.....	20
SNMPInterface.....	21
nIfIndex.....	22
nObjectId.....	23
strMacAddr.....	24
StrType.....	25
SNMPVlanFromDescription.....	26
The condition – description starts with vlan.....	27
strVlanId.....	28
The rest of the discovery-items and rules.....	29
<b>External configuration file</b> .....	<b>30</b>
<b>How to run the inventory</b> .....	<b>31</b>
Command line execution.....	31
Running using the GUI.....	31
Viewing the discovery log.....	31
<b>Final Page</b> .....	<b>32</b>

More Information..... 32  
Contact Us..... 32  
Sales..... 32  
Support..... 32

# Preface

---

This chapter provides an introduction to the structure and assumptions of this guide.

## The Purpose of This Guide

This guide contains a detailed explanation about one of the discovery rules examples supplied with the discovery engine samples.

## 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 the explained sample and its goals.
- A Path over the relevant 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

---

The discovery engine is mostly used to populate inventory tables based on data found within the organization. The product enables the user to access this distributed data via multiple sources and arrange it to its continece in an SQL based database, without programming and database definitions.

The process of defining the discovery rules is iterative, and dynamic. The rules usually change over time as new requirements are added, and old ones are found to be less productive.

The example we will give in this book, is related to IP network devices information collection.

We want to collect, using SNMP, multiple fields from devices such as switches and routers.

## Assumptions

We assume that the reader already read the “dicoverly.pdf” book. This book gives a full reference about the way to define discovery rules and the structure of the discovery rules.

It is also very recommended for the reader to read the “GUI programmers guide”, specially if there is a need to alter the GUI and add new reports to fit the newly discovered objects.

## The discovered environment

We have a list of switches and routers we want to discover.

In an Excel file, we have the devices IP addresses and their SNMP community.

After exporting the Excel to a CSV (Comma Separated Vector) file, we assume that the file looks like this:

```
strIpaddr,strCommunity
10.0.0.254,public
10.0.1.254,public
....
10.0.100.254,public
```

## What do we want to discover

We expect the discovery rules to loop on the list of devices given and for each to return the following data:

- The general system mib information containing (among others) the following fields:
  - Name
  - SysObjectId
  - SysLocation
  - SysDescription
- The Interfaces table of each device where we will extract (among others) the following fields:
  - IfIndex
  - IfDescription
  - IfName
  - ...
- The Vlan Table when relevant
- The Arp Table (if exists)

## The output of the discovery

We put the extracted fields in several tables:

### ***tblObject***

This table contains the information about the devices and their internal structure. It contains information about the nodes, interfaces and IP addresses of the devices.

### ***tblArp***

Contains information about the discovered ARP entries in the devices and the mapping of IPs to MAC addresses.

### ***tblVlanInfo***

This table contains information about vlans found.

## How to view the data

The product comes with pre-made reports to extract most of the data retrieved, however the user may easily add/alter reports in the product.

The user can simply add new options of reports to the menus, and , and can easily add new panels to query and show the report results by changing and adding configuration files to the product.

 The panels and reports added to the GUI, are also available for uses using the Web, see the products documentation for more details

---

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.

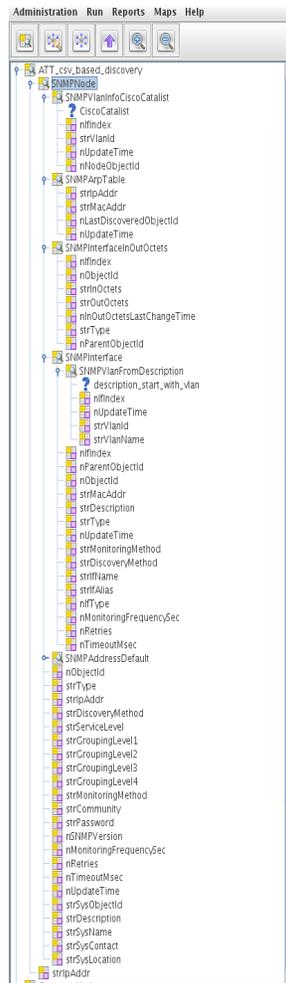
# Explanation of the Sample

---

## A short reminder about defining discovery rules

- Discovery rules are built from a list of discovery items cascaded in a tree hierarchy.
- A discovery item can populate a line or a set of lines
- The fields discovered in a discovery item up in the tree hierarchy can be used by discovery item found downwards in the tree
- All discovery items are XML files. They are located in [installdir]/conf/discoveryrules directory

## A general description of the discovery items within the example



A discovery process starts with a top discovery item that is attached to sub discovery items, each containing multiple fields :

- ATT\_csv\_based\_discovery – This is the sample's top discovery item. it is used to extract the list of nodes to discover and for each node its SNMP community.
  - SNMPNode - This discovery item is the one responsible for collecting the SNMP data from each devices. It collects & fills the global information about the nodes, and calls other discovery items for additional
    - SNMPVlanInfoCiscoCatalist – this discovery item extracts vlan information from the device using the cisco catalist private mib
    - SNMPArpTable – this discovery item extracts the ARP table of the device
    - SNMPInterface – is called by SNMPNode and extracts information related to the interfaces of the device
      - SNMPVlanFromDescription – will extract vlan information if the ifDescription starts

with the letters Vlan (as many devices conform).

- SNMPAddressDefault – builds an address record for the devices based on the queried node address
  - SNMPAddressDefaultCalcSubnetAddress – this discovery item, calculate the subnet address from the given Ip Address and store it in the table
- SNMPAddressTable – builds an address record for the devices based on the snmp query to the device's mib.
  - SNMPAddressTableCalcSubnetAddress – this discovery item, calculate the subnet address from the given Ip Address and store it in the table

## ATT\_csv\_based\_discovery

The screenshot shows a configuration window for a discovery rule named 'ATT\_csv\_based\_discovery'. The fields are as follows:

- Name:** ATT\_csv\_based\_discovery
- Description:** (Empty text area)
- Path:** /opt/jilroy/r/network/discovery/conf/discoveryrules/ATT\_csv\_based\_discovery
- Insert data into table name:** table
- Table Description:** (Empty text area)
- Item Type:** Table
- Parent Name List(CSV):** (Empty text area)
- Is Root Item:** 1
- Skip results:** 1
- Update record only when the following fields change:** 0
- List of fields to check (Comma separated):** (Empty text area)
- SQL Delete command:** (Empty text area)

The things to learn from this properties panel are:

- The main rules extract multiple lines as can be seen from the “Item Type” value of table.
- The data from this extraction will only be used as parameters for the down-wards discovery items (SNMPNode) and they wont be entered to the database as can be learned from “skip results” = 1

This discovery item, has only one discovery field named “strIpAddr”

## **stripAddr**

This is the properties screen for the stripAddr field item.

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:	stripAddr
Table field description:	
Unique field check SQL query:	
URL:	@ATTList@
separator:	,
User:	
Password:	
SQL Command:	select stripAddr, strCommunity FROM #TEMPTABLE#
Query field name:	stripAddr
Is first line in file, a header line:	1
Key field name in retrieved table (relevant only for table type items):	stripAddr

The things to learn from this discovery field are:

- We use the CSV data-source to read the CSV file given as a parameter @ATTList@.  
(Parameter passing will be explained later)
- We refer to the csv file as an SQL table, where we extract fields from the table, when the title line of the csv file acts as a field name mapper.
- We extract from each line both the IP and the community as we need them only as parameters that will be later used by the SNMPNode discovery item.
  - If we had to insert them both to the table using this discovery item, we would have had to define a new discovery field for the strCommunity field.
- We specified that the first line of the csv file should be skipped.

Now the logic of the discovery process calls for each line found all the discovery items that are below our main discovery item “ATT\_csv\_based\_discovery”

## SNMPNode

This discovery item contains many fields and many sub discovery items.

The screenshot shows a configuration window for a discovery rule named "SNMPNode". The interface is divided into two main sections: a left sidebar with labels and a right main area with input fields. The labels on the left include: Name, Description, Path, Insert data into table name, Table Description, Item Type, Parent Name List(CSV), Is Root Item, Skip results, Update record only when the following fields change, List of fields to check (Comma seperated), and SQL Delete command. The corresponding values in the right section are: Name: SNMPNode; Description: (empty); Path: /opt/jilroy/rinetworkdiscovery/conf/discoveryrules/BasicNetworkDiscovery/SNMPNode; Insert data into table name: tblObject; Table Description: (empty); Item Type: Line (dropdown menu); Parent Name List(CSV): Layer2\_static\_mapping,ATT\_csv\_based\_discovery,ATT\_csv\_based\_discovery; 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); SQL Delete command: (empty).

This discovery item is the one responsible for the discovery of SNMP data on each of the nodes.

Important things to learn:

- The data discovered is entered to a table named tblObject, as can be seen from the property: insert data into table name: tblobject

- The data will add only a single line. This can be learned from the Item Type=Line. The discovery items bellow it will add multiple records (SNMPInterface,...)
- The Path fields point to the location of the XML containing the definitions of this discovery item.

This discovery item has many discovery fields that will define the fields of the table.

**i** When you add a new discovery field it is automatically added to the table specified in the discovery item. However if you remove a discovery field, it will not remove this field from the table. You will have to get into the administration menu, and in the discovery rules management sub-menu remove the field from the table. Make sure that no other discovery rule uses this field.

In this book we will list only some of them. The others are defined in a similar way.

### ***nObjectld***

This field item defines the unique id of the tblObject for the 'node' entry identifying the device.

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:	nObjectld
Table field description:	
Unique field check SQL query:	<pre>SELECT distinct nObjectld 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:	

The important things to learn from this definition are:

- The field is a key field in the table. As such we have to check if it already exists and if yes then the record will be updated, otherwise a new record will be added. This is learned from the value of: is field a key = 1
- The Value of the field is automatically generated if none exists. This is learned from the Method value of AutoGenerateInt. This is a method supplied by the software that ensures uniqueness.

- The Unique Field check SQL query, ensures that there is no entry already for this node based on the IP address of the Node. The SQL :

```
SELECT
distinct nObjectId
FROM tblObject
WHERE
strType = 'Address' AND
strIpAddr = '@strIpAddr@'
```

searches if an 'Address' entry exists with the given IP address. Every key field must have such a query.

- It can be seen that the IP address in the query is a parameter (@strIpAddr@) that came from a parent discovery item.

### **strType**

This field item puts the constant 'Node' in the field strType. This field is not a key and its value generation method is 'Direct' which means taking the value found in the 'Value:' field and in this case not activating any function on it.

**i** The 'Value' field as all the fields accept parameters as their value. For example the field 'strIpAddr' under this node discovery-item is also a discover-field with a method of 'Direct' and a value of "@strIPAddr@".

This Field logic is the same as many other field-items under SNMPNode (strIpAddr, strDiscoveryMethod, strServiceLevel,...)

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:	strType
Table field description:	
Unique field check SQL query:	
Value:	Node
Value function:	None
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

## ***strSysObjectId***

This field-item extracts the sysobjectid from the node using SNMP

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:	3
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:	

The important things to notice in this properties panel are:

- The data generation method of this field is SNMP as can be learned from the 'Method' field.
- This field is not a key field
- The IP address and the community string used to extract the value are taken from the parameters extracted by the discovery-item above
- the SNMP Version used for this query is SNMP v1
- The Timeout value when we wait for a reply from the device is 3 sec
- The Retry count in case the device does not answer is 3

- The Maximum reply length to the SNMP query is 64K bytes
- The OID field specified the one for the SysObjectId
- The OID description is used only for documentation reasons and may be left empty, although it is recommended that it would stay.
- If an SNMP error has already occurred for the device, we do not try again in this discovery cycle to access the device. This saves time in the discovery cycle when the device does not respond.

This field is very similar to the strSysDescription, strSysName, strSysContact,...

### ***nUpdateTime***

This discovery-field gets the current time as a value. This is a 'Direct' Method field-item. It returns the number of seconds from 1.1.1970

Method (save if change to get new parameters):	Direct
Field Type:	Int
Is field a key:	0
Is field a table key (For Table Discovery Items):	0
Table field name:	nUpdateTime
Table field description:	
Unique field check SQL query:	
Value:	
Value function:	CurrentTime0
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

The value is automatically generated by the Value Function 'CurrentTime()'

# SNMPInterface

This discovery item fills the database with multiple records, one for each interface of the device.

It contains many fields and many sub discovery items.

<b>Name:</b>	SNMPInterface
<b>Description:</b>	
<b>Path:</b>	/opt/jilroy/rinetworkdiscovery/conf/discoveryrules/BasicNetworkDiscovery/SNMPInterface
<b>Insert data into table name:</b>	tblObject
<b>Table Description:</b>	
<b>Item Type:</b>	Table
<b>Parent Name List(CSV):</b>	SNMPNode
<b>Is Root Item:</b>	0
<b>Skip results:</b>	0
<b>Update record only when the following fields change:</b>	0
<b>List of fields to check (Comma seperated):</b>	
<b>SQL Delete command:</b>	

The important things to learn from this properties panel:

- This discovery item fills a number of records (table) opposed to the SNMPnode that filled only a single line. This can be learned from the value of 'Item Type: Table' . This point is very important as tables are defined in a different way than lines. All the field-items of a table must either return multiple lines and in this case, they must return 2 values when the first is the actual field of the table, and the second one is the value of the key field for the multi line values returned.
- The data will be put in tblObject.

**i** The value of the field 'Insert data into table name:' could be left empty and in this case the value would have been taken from the parent discovery-item.

This discovery-item contains many discovery-fields. We will list a few of them that are different from the ones we already explained.

### ***nIfIndex***

This discovery-field returns **all** the ifIndex values of the interfaces of the node. Its discovery method is 'SNMP' This field is the key value of the IF-table in the mib. This means that the ifIndex is returned as part of the key in the reply to SNMP request.

Method (save if change to get new parameters):	SNMP
Field Type:	Int
Is field a key:	0
Is field a table key (For Table Discovery Items):	1
Table field name:	nIfIndex
Table field description:	
Unique field check SQL query:	
IP Address:	@stripAddr@
Community:	@strCommunity@
SNMP Version (1 2 3):	1
Timeout (msec):	3000
Retry count:	3
Max PDU Size:	64000
OID:	1.3.6.1.2.1.2.2.1.1
OID description:	mib2.interfaces(2).ifTable(2).ifEntry(1).ifIndex(1)
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)	0
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

Things to learn from this properties panel:

- This field is the key of the 'table' retrieved. This is learned from the value of the field: 'is field a table key' = 1
- Although we could have taken the value of the field from the key returned by the SNMP query we still take it from the value as specified in 'Get key value of snmp reply: 1'

- The other fields are the same as in the previous SNMP method discovery fields we encountered before

**i** The table-field must be the first field defined in the table.

### ***nObjectId***

This discovery-field acts as a key for the record in the tblObject. It is similar to the field nObjectId in the SNMPNode discovery-item, however there is a major difference. It is generated for every line that is created using this discovery-item.

Because of that when we check if there is already an entry for the interface, we have to return also the key of the records defined by this discovery-item in addition to the key of the global table.

This is why the SQL selection of the 'Unique field check SQL query:' returns 2 fields:

- The global key field
- The SNMPInterface table key which is the nIfIndex field value.

This way the software can identify the connection between the current field and the table key field for each value.

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 nObjectId,nIfIndex FROM tblObject WHERE strType = 'Interface' AND nParentObjectId = @nObjectId@</pre>
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

Additional things to learn from this properties panel:

- This is a global key field in the table. This can be learned from the value: 'is field a key=1'
- The 'Unique field check SQL query' was explained above

### **strMacAddr**

This discovery-field extracts all the mac addresses found in the IF table. As the IfIndex is the key of this Mib table, it is returned with the reply for the Mac addresses field in the table. This is how the product knows how to associate each mac with each IfIndex.

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:	strMacAddr
Table field description:	
Unique field check SQL query:	
IP Address:	@strIpAddr@
Community:	@strCommunity@
SNMP Version (1 2 3):	1
Timeout (msec):	3000
Retry count:	3
Max PDU Size:	64000
OID:	1.3.6.1.2.1.2.2.1.6
OID description:	mib2.interfaces(2).ifTable(2).ifEntry(1).ifPhysAddress(6)
Can data be in Hex format:	1
Get key value of snmp reply:	0
Skip Protocol for node on failure(Valid only for Line Items)	0
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

The things to learn from this properties panel are:

- We found that the Mac address in some devices is held as binary data. This is why we specified that the data can be in hex format. The output will be a hex representation of the address only if it contains non printable letters
- We did not specify the table key that matches the mac address, as it is automatically returned in the snmp query as part of the reply. This means that the key value for such tables must be defined exactly as the key field in the SNMP mib definition of the device.

## StrType

This discovery-field sets the value of 'Interface' to all the records added by this discovery-item. (regardless of the nlfIndex of the record added). This is why in this case we did not have to specify the key of the record to it the data belongs.

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:	strType
Table field description:	
Unique field check SQL query:	
Value:	Interface
Value function:	None
Final SQL Query:	
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

# SNMPVlanFromDescription

This discovery-item extracts the vlan id and name from the ifDescription field, which was entered to the strDescription table field in tblObject. The Discovery-item is call for each interface with the specific information selected for the specific interface.

<b>Name:</b>	SNMPVlanFromDescription
<b>Description:</b>	
<b>Path:</b>	/opt/jilroy/rinetworkdiscovery/conf/discoveryrules/SNMPVlanFromDescription
<b>Insert data into table name:</b>	tblVlanInfo
<b>Table Description:</b>	
<b>Item Type:</b>	Line
<b>Parent Name List(CSV):</b>	SNMPInterface
<b>Is Root Item:</b>	0
<b>Skip results:</b>	0
<b>Update record only when the following fields change:</b>	0
<b>List of fields to check (Comma seperated):</b>	
<b>SQL Delete command:</b>	

Things to learn from this properties panel:

- The output of this discovery-item is a line, as can be learned from the field 'Item Type: Line'
- The output of this line is targeted to a table named tblVlanInfo

The new thing about this discovery-item is that it contains a condition that checks if this discovery-item should be executed with the given input data.

### ***The condition – description starts with vlan***

This condition checks if the description field extracted by the SNMPInterface discovery-item starts with the string vlan. If it does this means that we can extract the vlan information from the this field.

The screenshot shows a configuration window for a discovery rule condition. The window has a light gray background and a white content area. On the left side, there are labels for 'Method (save if change to get new parameters):', 'Name:', 'Key Name:', 'Value:', and 'Value Function:'. The right side contains the corresponding input fields. The 'Method' dropdown is set to 'Fields\_Compare'. The 'Name' text box contains 'description\_start\_with\_vlan'. The 'Key Name' text box contains '@strDescription@'. The 'Value' text area contains 'Vlan'. The 'Value Function' dropdown is set to 'KeyStartWithValue'.

Here we apply the function: KeyStartsWithValue to the value of strDescription for the interface called.

## ***strVlanId***

This discovery-field extract the vlan Id from the description field.

Method (save if change to get new parameters):	Direct
Field Type:	String
Is field a key:	1
Is field a table key (For Table Discovery Items):	0
Table field name:	strVlanId
Table field description:	
Unique field check SQL query:	
Value:	@strDescription@
Value function:	None
Final SQL Query:	select top 1 substr('@strDescription@',5) from tblObject
Final SQL Selected field name:	
Final SQL Key field name in retrieved table:	

Things to learn from this properties panel:

- This field-item defines the key of the table.
- The Method used for this field is Direct
- In this Discover-field we use for the first time the Final SQL query and we use it in order to manipulate the result with SQL methods. Here we cut the value of the description which in this case is in the format of *Vlannumber for example: Vlan10*. We extract the value 10 from the string and this is the field value.

## The rest of the discovery-items and rules

The rest of the discovery-items and fields are similar to the ones we have already seen.

The user can easily add new discovery items and under the root discovery-item, or can remove them. Discovery items can fields can be copied/moved to needed locations.

It is recommended to have a development environment where tests will be done, there the user will be able to clean the database after every run either by recreating the database (by running the command: [installdir]/conf/dbconf ) or by using sql to truncate the tables on which he works on.

# External configuration file

---

We have seen that some of the fields in the discovery-items defined use input that was passed as parameters. A parameter is defined by a string in the following format @parm\_name@.

Some of the parameters are fields that were discovered in discovery-items up in the tree, however there are others that were not discovered previously.

In order to supply those parameters, we give the discovery program an input file that contains the value of these fields.

Following is the content of such a file:

```
ATTList=file:///work/ips_to_discover.csv
STRDISCOVERYMETHOD=SNMP
STRSERVICELEVEL=
STRGROUPINGLEVEL1=
STRGROUPINGLEVEL2=
STRGROUPINGLEVEL3=
STRGROUPINGLEVEL4=
STRMONITORINGMETHOD=
STRCOMMUNITY=PUBLIC
STRPASSWORD=
NSNMPVERSION=1
NMONITORINGFREQUENCYSEC=0
NRETRIES=3
NTIMEOUTMSEC=5
NSNMPGETPORTID=10000
```

You should use these parameters in order to define global parameters that you do not want to hard-code.

This sample file is found in [installdir]/conf/ATT\_csv\_based\_discovery\_sample.ini

# How to run the inventory

---

To run the discovery rules defined, you can use one of to methods.

## Command line execution

Use the following command to run

```
[installdir]/bin/runinventory ATT_csv_based_discovery path_to_ini_file
```

## Running using the GUI

You can launch the inventory using the GUI by right clicking the discovery-rule and selecting from the popup menu the option: Run Inventory

After selecting this option, a pop-up will appear requesting you to give the path to the .ini file. After selecting the file, the Inventory program will start.

## Viewing the discovery log

Currently the Inventory program is defined in a way that prints a detailed log.

The log is found in the [installdir]/log/inventory.log file.

This log gives a very detailed description of the progress in the discovery process. You can understand from it, what was exactly done, and what problems were encountered.

It is very important to understand the structure and flow of this log file.

# 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)

### Support

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

FTP Site

All Jilroy products can be downloaded from our Web site.