



# **Map Generator 3.0**

## Mapping generation Reference Guide



**Jilroy Software LTD 2009**

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 © 2000-2009 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 NT® is a US registered trademark of Microsoft Corporation.

Windows® 2000 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>6</b>
<b>Introduction.....</b>	<b>8</b>
The mapping language.....	8
A Symbol.....	8
A Link.....	8
A Map.....	8
A child map.....	8
A topological view.....	8
A Map Layout.....	9
Symbol color.....	9
Why do we need topological views.....	9
Hierarchical IP Network maps.....	9
Server farm map.....	9
State/Color propagation.....	10
<b>Defining Mapping Rules.....</b>	<b>12</b>
General Description.....	13
Mapping Generators Items Hierarchy.....	13
A schematic view of Mapping Items relationship.....	14
Mapping rules definition GUI.....	15
The wizard general functionality.....	15
View Generator Object.....	16
Mapping items hierarchy.....	16
Mapping Items parameters.....	16
View Generator properties panel.....	16
View Generator properties.....	17
Name.....	17
Label.....	17
Layout.....	18
Background Image Path.....	18
Icon full path.....	18
Run Command Prior to generation.....	18
Re-execute Frequency.....	18
Symbols Generators.....	20
Common Symbols Generators Item properties.....	20
Method.....	20

Name.....	21
Label.....	21
Symbol Type.....	21
Symbol Subtype.....	21
User Data.....	21
Initial state.....	21
State Propagation Weight.....	21
Layout.....	22
Background Image Path.....	22
Icon full path.....	22
Container Symbol Generator.....	23
CSV Symbols Generator.....	24
URL.....	25
User.....	25
Password.....	25
SQL command.....	25
Query field name.....	25
Is First line a header.....	25
SNMP Symbols Generator.....	26
IP Address.....	26
Community.....	27
SNMP Version.....	27
Timeout in milliseconds.....	27
Retry count.....	27
Max PDU size.....	27
OID.....	27
OID description.....	27
Can data be in hex format.....	27
SQL command.....	27
Query field name.....	27
SQL Field item.....	28
SQL Driver name.....	28
SQL Connection string.....	28
User.....	28
Password.....	28
SQL command.....	29
Query field name.....	29
The Link Generators.....	30

Common Links Generators Item properties.....	30
Method.....	30
Name.....	30
Label.....	30
Symbol Type.....	31
Symbol Subtype.....	31
User Data.....	31
State Propagation Weight.....	31
Layout.....	31
Background Image Path.....	31
Icon full path.....	31
'Link ALL' Links Generator.....	33
From Symbols Generator name.....	33
To Symbols Generator name.....	33
'Link SQL' Links Generator.....	34
SQL command.....	34
<b>Final Page.....</b>	<b>35</b>
More Information.....	35
Contact Us.....	35
Sales.....	35
Product Management.....	35

# Preface

---

Welcome to Map Generator 3.0 Reference guide. This tool enables you to define the rules that will ultimately generate topological maps under the Jilroy's products GUI. 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 the Map Generator 3.0 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 Jilroy GUI containing the Map Generator capability.

## 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 Map Generator 3.0 and its goals.
- Defining Mapping rules – A description of the how to define mapping rules that will ultimately be converted to topological maps.
- 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 Map Generator 3.0 from Jilroy Software is a very powerful generic tool used to automatically generate topological views based on Mapping rules defined via this tool. It can be used to generate topological maps for every possible need. Possible uses are:

1. Hierarchical IP Network maps
2. Server farms maps
3. Business topology maps

## The mapping language

In this section we will define the language and wording that we will use in the rest of this document.

### A Symbol

An icon that appears on a single windows of the GUI. Usually a symbol represents an object in the world that is mapped. For example:

- A Building
- A room
- A computer
- an Application

### A Link

A line connecting between two symbols

### A Map

A set of symbols that appear in a single window of the GUI.

### A child map

A map that appears when drilling down through a symbol or a link on a given map

### A topological view

A set of maps & child map that together visualize the data their creator wanted to display

## A Map Layout

A map layout determines how the symbols are distributed within a given map.

## Symbol color

A symbol can have a color which represents visually its state. For example:

- Red – Fatal problem
- Yellow – Warning state
- Green – No problems

## Why do we need topological views

There is a say that a single picture is worth 1000 words. Topological maps are one instance of such a picture. They are perfect to visualize hierarchal constructs, which are composed of objects in different levels of detail or abstractions.

Following is a list of examples of such hierarchal views. The list given is a possible topological hierarchy , however many other hierarchies may exist

### Hierarchical IP Network maps

- Sites – the geographical distribution of the enterprise – with links defining the connections between the sites
- Site backbone – the major routers and backbone switches that exist in a given site with the connections between them.
- Backbone switch – the map that includes all the second level switches connected to the backbone switch
- Switch map – the map that includes the switch and all the nodes that are connected to a given switch
- Interfaces map – the map that lists all the interfaces of the selected node.

When double clicking a symbol at a given level you will see the sub map related to it. For example when you double click a backbone switch you will see a map containing all the second level switches connected to that specific backbone switch.

### Server farm map

- Server farms locations – a geographical distribution of the server maps

- A business grouping of the servers – for example finance, CRM, ERP, ...
- Business servers grouping – the list of all servers within a given group
- Services – the list of the major services within a given server, for example: Database

## State/Color propagation

The topological maps supplied have the ability to propagate states/colors from a child to its parent. This options enables drilling down toward a problem, relatively very fast.

The product allows the user to determine what the importance of a given object to the color of its parent by giving a weight to a child, and giving thresholds to colors on nodes.

The status of a node can be set by external events such as:

- The result of a status check using ICMP/SNMP or other mechanisms
- The result of receiving a trap
- Based on external logic

There are command lines that enable the user to change a state of an object from other products.

## Advantages of the product

- The product allows you to define rules that control the exact structure of the topological views, and the relationship between elements in the maps. All this is done using wizards and without any programming.
- The product can connect to any data source in order to extract data that will be used later to build the topological maps.
- The product is one of the most flexible mapping products existing in the market today. You as the customer have control on almost every thing displayed. You can define what will be displayed when pressing an icon, and what will be the popup menu for that element for every icon.
- The product runs 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.

Jilroy Software encourages you to use the product and extend it, by defining your own mapping 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 Mapping Rules

---

The Mapping Generator purpose is to allow the user to define, using a wizard and similar to the way reports are defined in a report generator, the format of topological views that represent its inventory, in the best way, and integrate into the generated maps, information that is collected using monitoring technologies.

The user can build multiple topological views that either map different parts of its organization or maps the same objects in different aspects.

Usually the input source for building the topological maps is the the inventory database generated by the “Jilroy Inventory platform”. This enables the user to use all the protocols existing in the inventory platform as data sources for the maps construction. Listed below is a partial list of the discovery protocols supported by the “Jilroy Inventory platform”:

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
11. and more...

However the user can also connect to any other database or CSV file, and currently we also support using the SNMP data-source directly from the mapping generator.

There are 2 methods for defining mapping rules.

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

The mapping rules language is XML.

## General Description

A Map generation of a single Topological view is composed of:

1. Defining a View generator object , which contains the general instructions regarding the view, and information about the root symbol that represents the view.
2. Defining under that view, one or more symbol generators which define the symbols hierarchy . The symbol generators control all the aspects of representing an object on a map, from its icon, through its label, and up to the background image that appears on the sub-map that represents its children. There are rules generators that define links between symbols in a given sub-map.

The symbol generators format a structure of a tree when a parent symbol generator can have multiple child symbol generators, that in the generated map, will represent the sub-map of a given symbol.

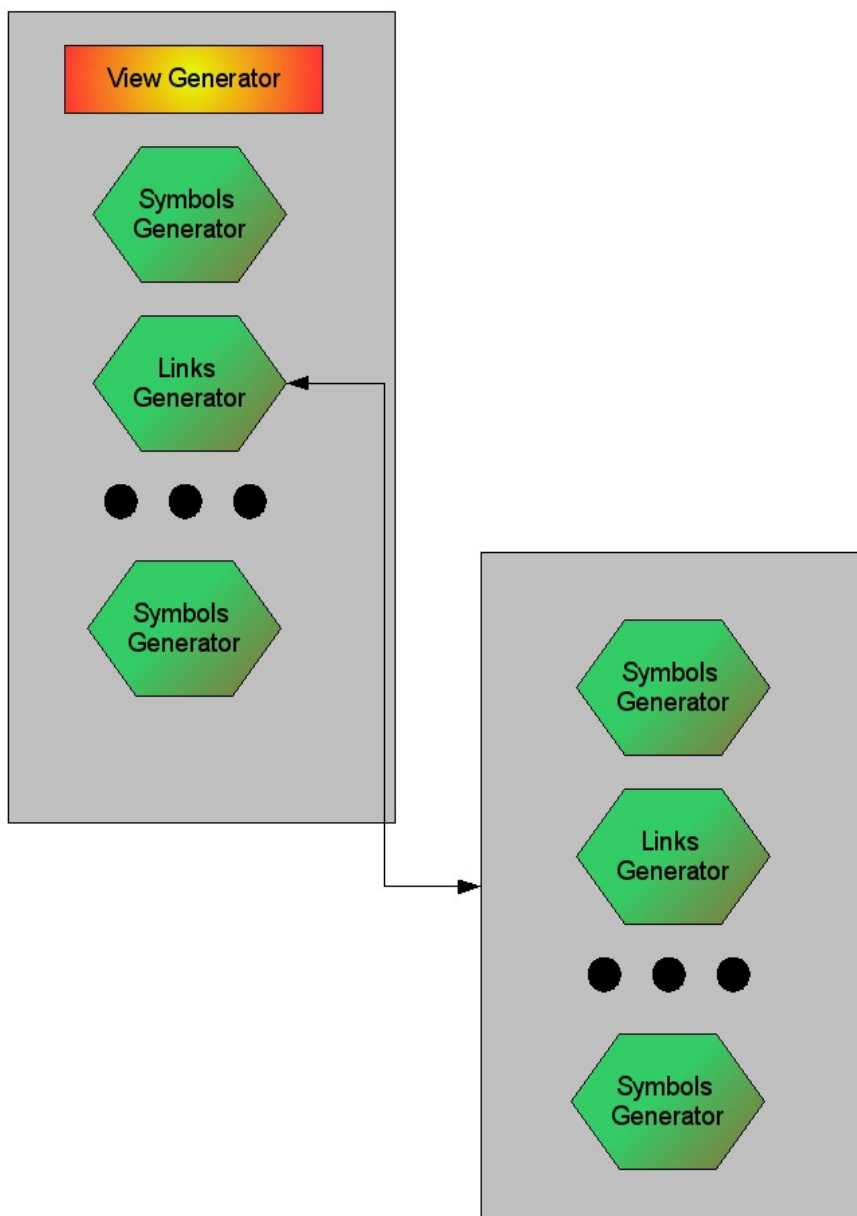
Please look at our site for new available mapping rules samples.

## Mapping Generators Items Hierarchy

Mapping Generators items are the building blocks of creating an active topological view. When we run a mapping generation job, we start with a root View Generator item which points to symbol generators.

In this section we explain schematically the relationships between Mapping Generators items.

## A schematic view of Mapping Items relationship

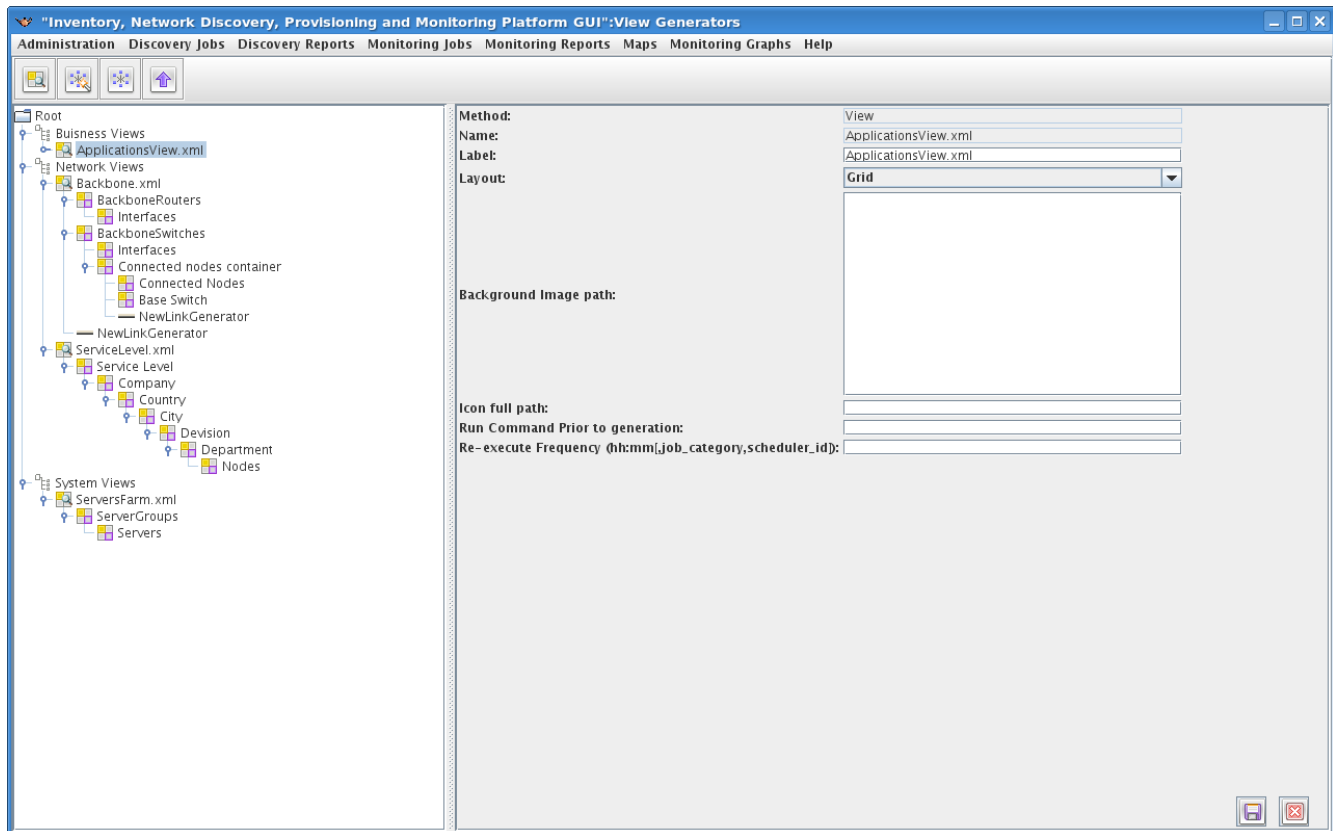


# Mapping rules definition GUI

To enter the mapping rules definition panel you should select:

**Administration->Mapping rules->Mapping rules definitions**

The panel opened is:



This panel is a wizard used to define mapping rules item sequences.

The panel is divided into 2 sections:

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

## 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.

## View Generator Object

The View Generator Object, is the main container of instructions on how to create a mapping hierarchy. It represents an XML in the "[INSTALLDIR/conf/viewgenerators/"] .

After generating the view it is represented as an icon in the root of the mapping menu option:

**Maps->All defined Maps**

### *Mapping items hierarchy*

The mapping process starts from a 'root' view generator, and then it processes all its children in an orderly manner. Each mapping Item can use all the information generated by its parent chain when extracting data that should be used in the mapping process (for example: for selecting symbols on a given sub-map, or creating a meaningful label).

### Mapping Items parameters



A mapping item, can use parameters that are passed to it by its parent, and in the case of view generator, by the launcher of the mapping process. The parameters are string based, and contain a key and a value. The parameters are referred in the child mapping items 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 mapping item, can be accessed as parameters by its children.

## View Generator properties panel

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

Method:	View
Name:	ApplicationsView.xml
Label:	ApplicationsView.xml
Layout:	Grid
Background Image path:	
Icon full path:	
Run Command Prior to generation:	
Re-execute Frequency (hh:mm[job_category,scheduler_id]):	



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 view Generator item.

## View Generator properties

The following section will list the properties used to define a view generator.

### ***Name***

This property represents the name of the view generator. It must be unique to all view generators. When prompted from the GUI, the name can include a path name relative to the [INSTALLDIR]/conf/viewgenerators/ directory if the user wants to create an hierarchy of view generators 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

### ***Label***

This property holds the label of the icon that represents the view. If no label is given then the name is taken as label, when the view is generated.

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

## ***Layout***

The layout of the sub-map that appears when this icon is double clicked when this view is generated. There is a set of predefined layouts, and there is an option that the user to arrange the icons distribution on its own.

Valid values are selected from the combo box

## ***Background Image Path***

This is a name of an image file that will act as the background image on the sub-map that will appear when the icon of the view is clicked.

Valid values are: a full path to a valid file name

## ***Icon full path***

A pointer to the icon file location. Icon samples can be found in the [INSTALLDIR]/resources directory.

Valid values are: a full path to a valid file name

## ***Run Command Prior to generation***

Sometimes it is needed to prepare data prior to running the view Generator, for example :in order to extract information from unsupported data sources.

This entry allows to specify a system command that will be executed prior to the generation of the view.

Valid values are: a full path to a valid file name

## ***Re-execute Frequency***

A view is usually dynamic, and changes s time passes. This parameter allows you to define how frequently the view is generated.

Each time the view is re-generated, new symbols appear are added to the existing views, and symbols that no longer match the generation rules are removed.

The format of this field is: hh:mm, job\_category, scheduler\_id

where:

**hh:mm**: specifies the frequency in which the view is generated.

**job\_category**: specifies the name of the scheduling category, as defined in [INSTALLDIR]/conf/categories.txt

refer to the scheduler documentation for an explanation on this option

**scheduler\_id:** the Id of the scheduler on which the view will run (by default there is only scheduler #1

# Symbols Generators

The Symbols Generator mapping item, represents a set of symbols that will appear on the sub-map that appears when double clicking a symbol generated by the parent symbol generator of this symbol generator.


There are currently a few types of symbol generators, that defer in their data extraction methods. Each has its own specific properties while there many common properties to all symbols generator items.

The supported extraction methods are:

- Container – A single symbol is generated by this generator.
- CSV – This method is used for generating symbols extracted from a CSV file. The data will be extracted using an SQL that will look at the CSV file translated as a Table.
- SQL – This method connects to a database and extracts data from it using SQL commands.
- SNMP – This method uses SNMP to extract either a single field or a table of fields, when SQL is used to filter the requested data.

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

---

 all the fields can get a value that uses parent parameters in the format of '@ParameterName@'

---

## Common Symbols Generators Item properties

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

### **Method**

The Symbols Generators method defines how the symbol content is calculated. Currently there are several methods supported, allowing to extract data from multiple data sources.

The Valid values are:

- Container
- CSV
- SQL
- SNMP

***Name***

A descriptive name of the symbols generator entry. This value will be used in some cases when no label was given

Valid Values: String

***Label***

The label that will appear near the generated symbol.

Valid Values: String

***Symbol Type***

A type value of the generated symbol. This is used if there is a need to select a popup menu for the generated symbol based on its type.

Valid values: String

***Symbol Subtype***

A sub-type value of the generated symbol. This is used if there is a need to select a popup menu for the generated symbol based on its sub-type.

Valid values: String

***User Data***

This is a field that can be used by the user for reference. We use it in the product as a way to select the symbol information when the user selects the “properties...” popup menu on the symbol.

***Initial state***

This defines the initial state of the symbol, when it is created.

Valid values: Look at the symbol coloring chapter for a description on the valid values

***State Propagation Weight***

This defines when propagating the state of this symbol what will be the symbol importance in calculating the parent state.

Valid values: Look at the symbol coloring chapter for a description on the valid values

## ***Layout***

The layout of the sub-map that appears when this icon is double clicked when this symbol is generated. There is a set of predefined layouts, and there is an option that the user to arrange the icons distribution on its own.

Valid values are selected from the combo box

## ***Background Image Path***

This is a name of an image file that will act as the background image on the sub-map that will appear when the icon of the symbol is clicked.

Valid values are: a full path to a valid file name

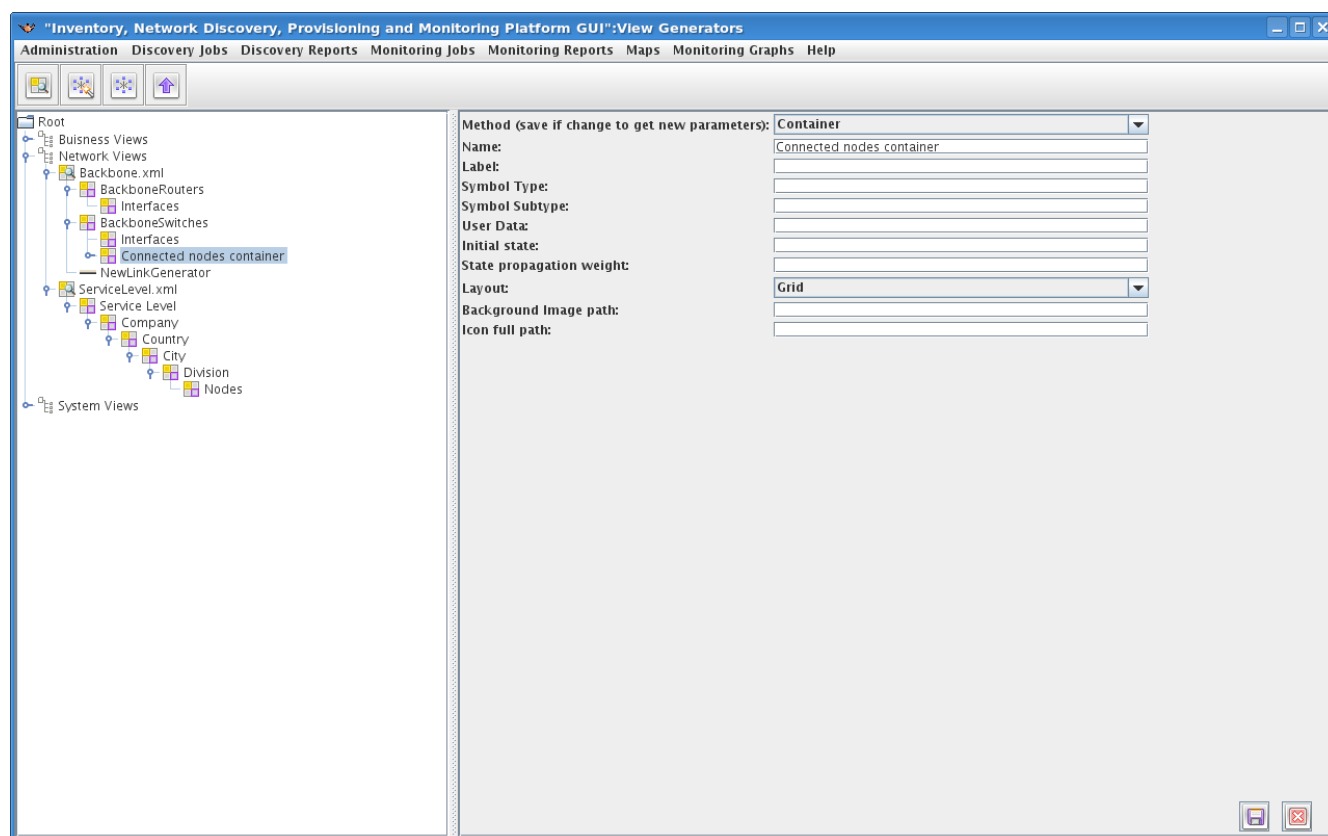
## ***Icon full path***

A pointer to the icon file location. Icon samples can be found in the [INSTALLDIR]/resources directory.

Valid values are: a full path to a valid file name

## Container Symbol Generator

The Container symbol generator, inserts a single symbol to the sub-map, and its name taken either from the label, and if not found for the entry name.

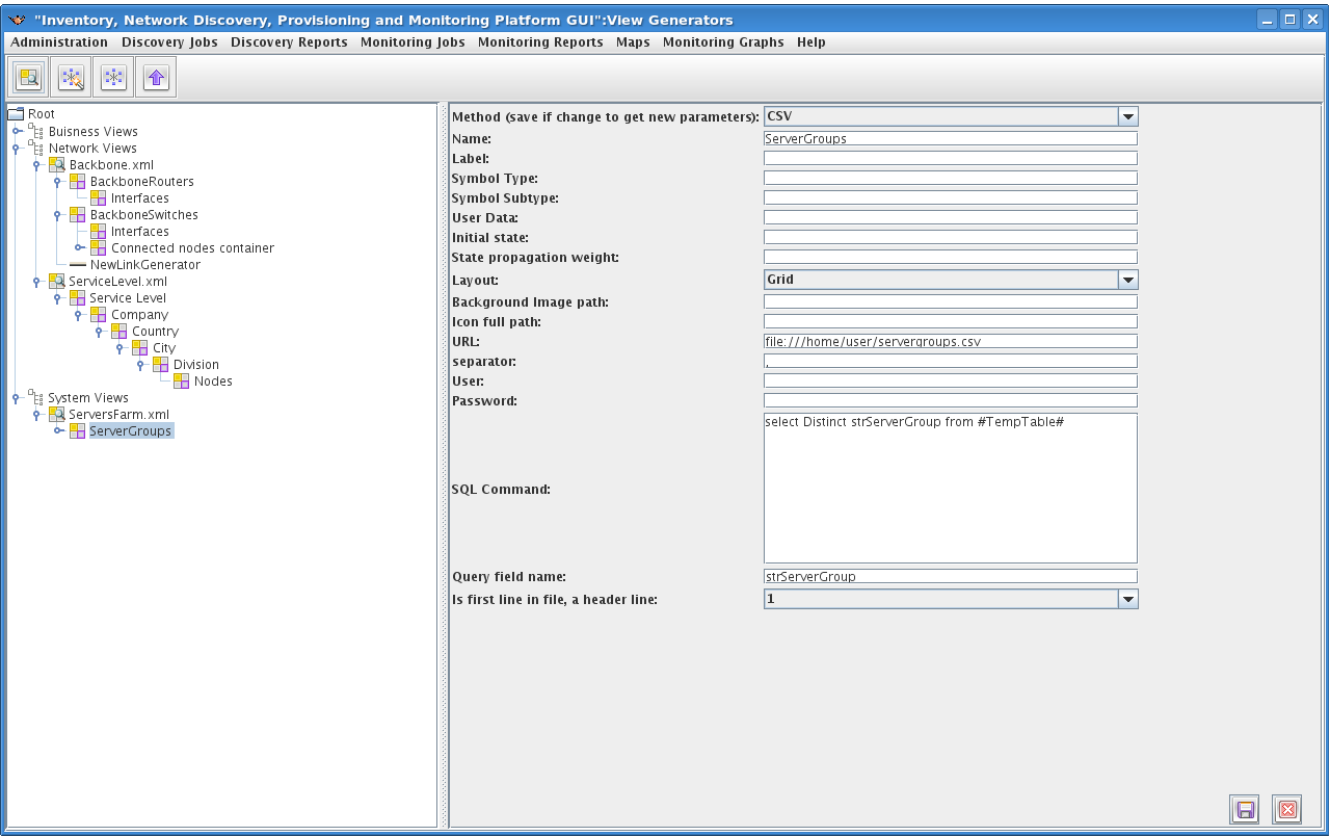


### Specific fields:

There are no specific parameters for this symbol generator type.

# CSV Symbols Generator

The CSV Symbols Generator, 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.



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 map generation 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,...

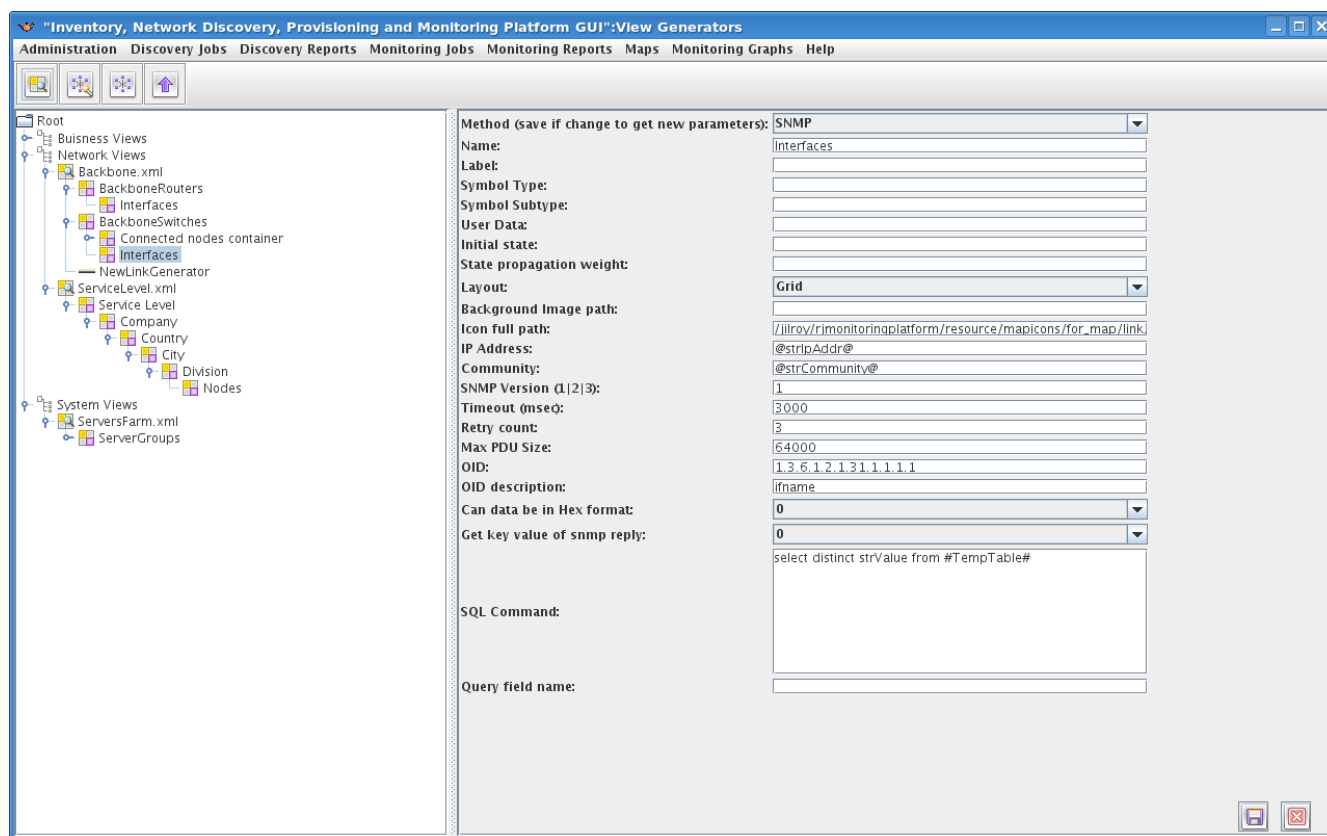
## SNMP Symbols Generator

This symbols generator generates symbols based on data located in the SNMP mib of the destination node.

From the results returned by the defined SNMP query, which is equivalent to 'snmpwalk' on the supplied OID, a temporary table is created with the following fields:

- strOID
- strValue
- strKey

Then the defined SQL query is executed on the results



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, although snmp v3 can be supplied upon request.

***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.

***SQL command***

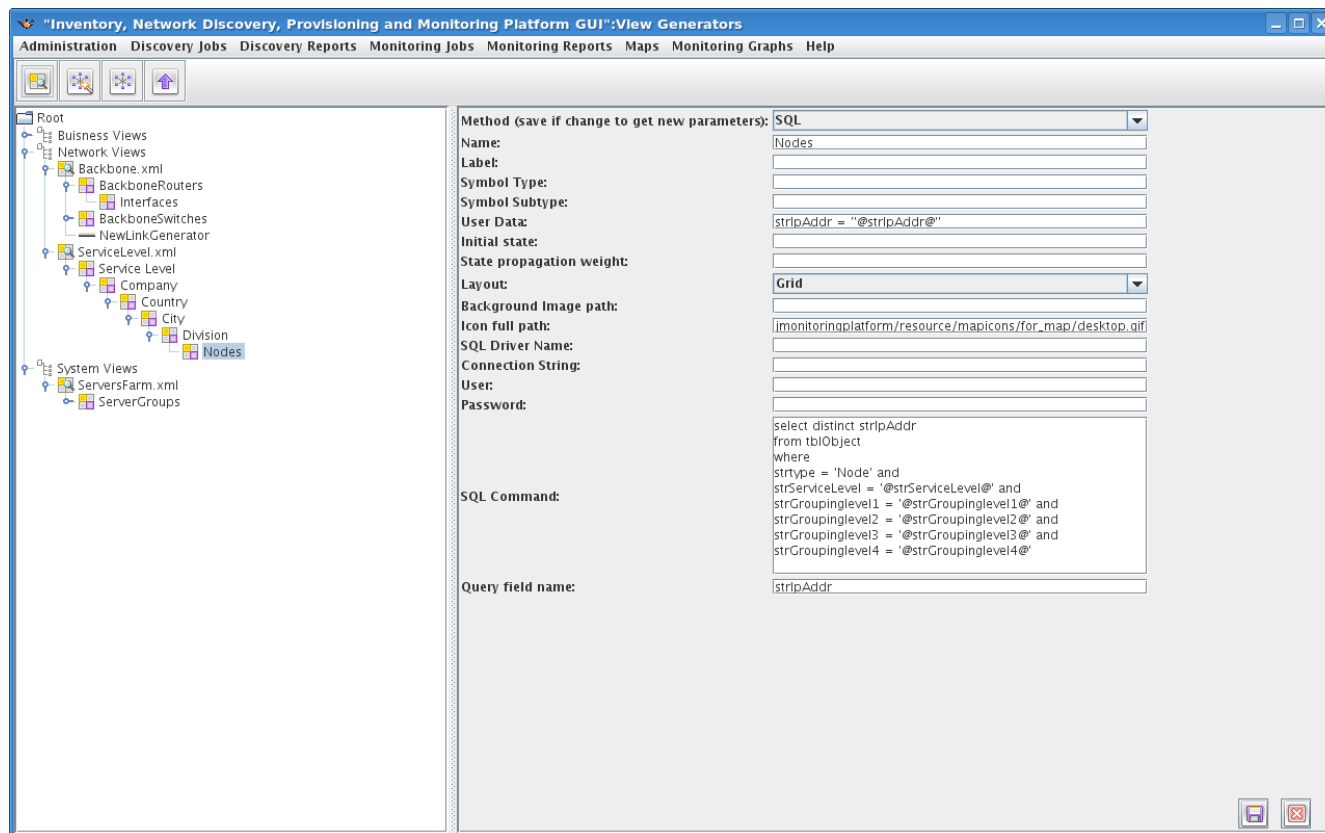
The SQL command that will be executed on the SNMP results content in order to extract the requested values

***Query field name***

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

## SQL Field item

This field item returns values located in SQL repositories.



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.

# The Link Generators

The link generators are used to generate links between symbols in a given sub-map.

Currently there are a few types of link generators supported. More are planned in the future.

- Link ALL – connects all symbols from a given symbols generators rule, to all symbols generated by another symbols generator.
- Link SQL – connects symbols on a given sub-map based on an SQL query that defines the relations between the symbols.

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

---

**i** all the fields can get a value that uses parent parameters in the format of '[@ParameterName@](#)'

---

## Common Links Generators Item properties

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

### **Method**

The Symbols Generators method defines how the symbol content is calculated. Currently there are several methods supported, allowing to extract data from multiple data sources.

The Valid values are:

- Link ALL
- Link SQL

### **Name**

A descriptive name of the links generator entry.

Valid Values: String

### **Label**

The label that will appear near the generated link.

Valid Values: String

## ***Symbol Type***

A type value of the generated symbol. This is used if there is a need to select a popup menu for the generated link based on its type.

Valid values: String

## ***Symbol Subtype***

A sub-type value of the generated symbol. This is used if there is a need to select a popup menu for the generated link based on its sub-type.

Valid values: String

## ***User Data***

This is a field that can be used by the user for reference. We use it in the product as a way to select the link information when the user selects the “properties...” popup menu on the symbol.

## ***State Propagation Weight***

This defines when propagating the state of this link what will be the symbol importance in calculating the parent state.

Valid values: Look at the symbol coloring chapter for a description on the valid values

## ***Layout***

The layout of the sub-map that appears when this icon is double clicked when this link is generated. There is a set of predefined layouts, and there is an option that the user to arrange the icons distribution on its own.

Valid values are selected from the combo box

## ***Background Image Path***

This is a name of an image file that will act as the background image on the sub-map that will appear when the icon of the link is clicked.

Valid values are: a full path to a valid file name

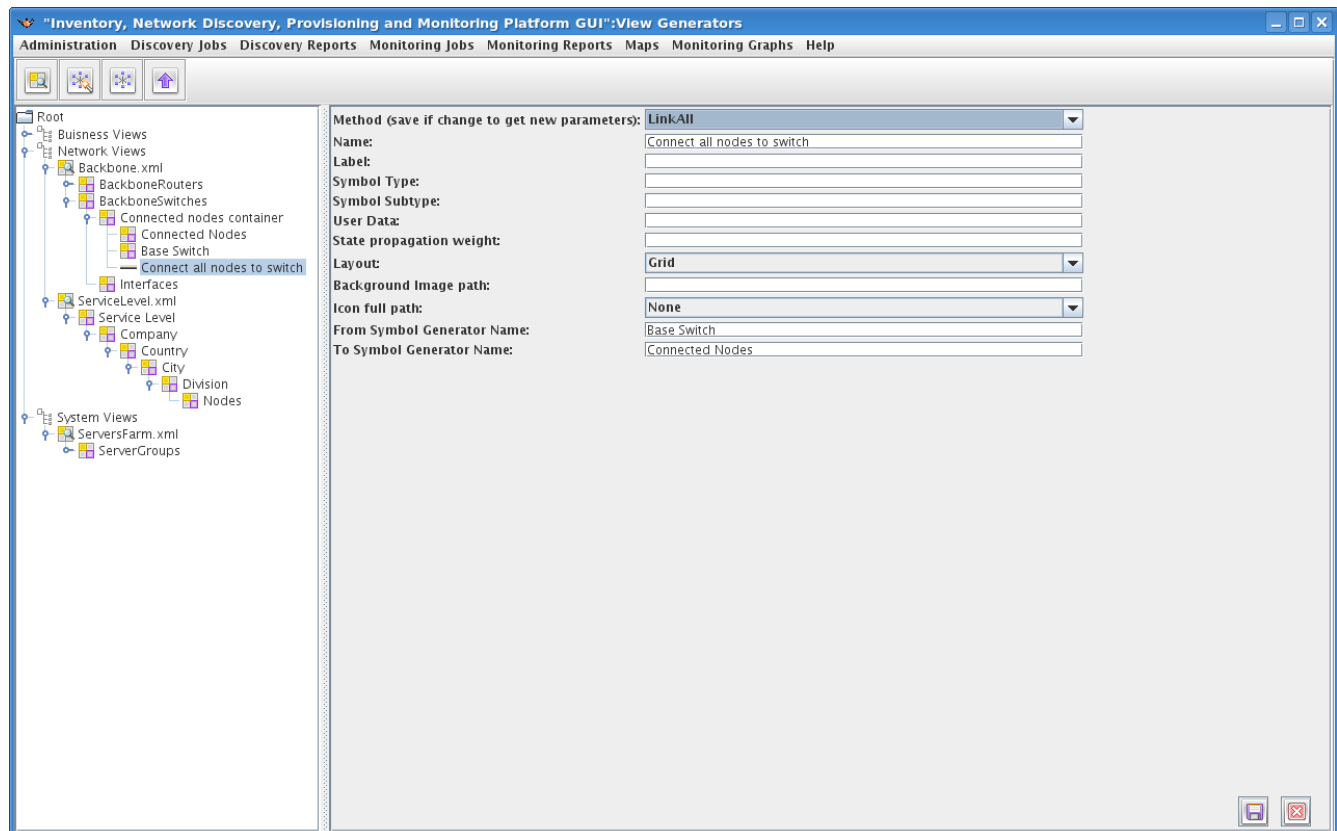
## ***Icon full path***

A pointer to the icon file location. Icon samples can be found in the [INSTALLDIR]/resources directory.

Valid values are: a full path to a valid file name

## 'Link ALL' Links Generator

The 'Link ALL' links generator connects all symbols generated by a given symbols generator with all symbols generated by an other given symbols generator. All symbols must be in the same sub-map.



Specific fields:

### ***From Symbols Generator name***

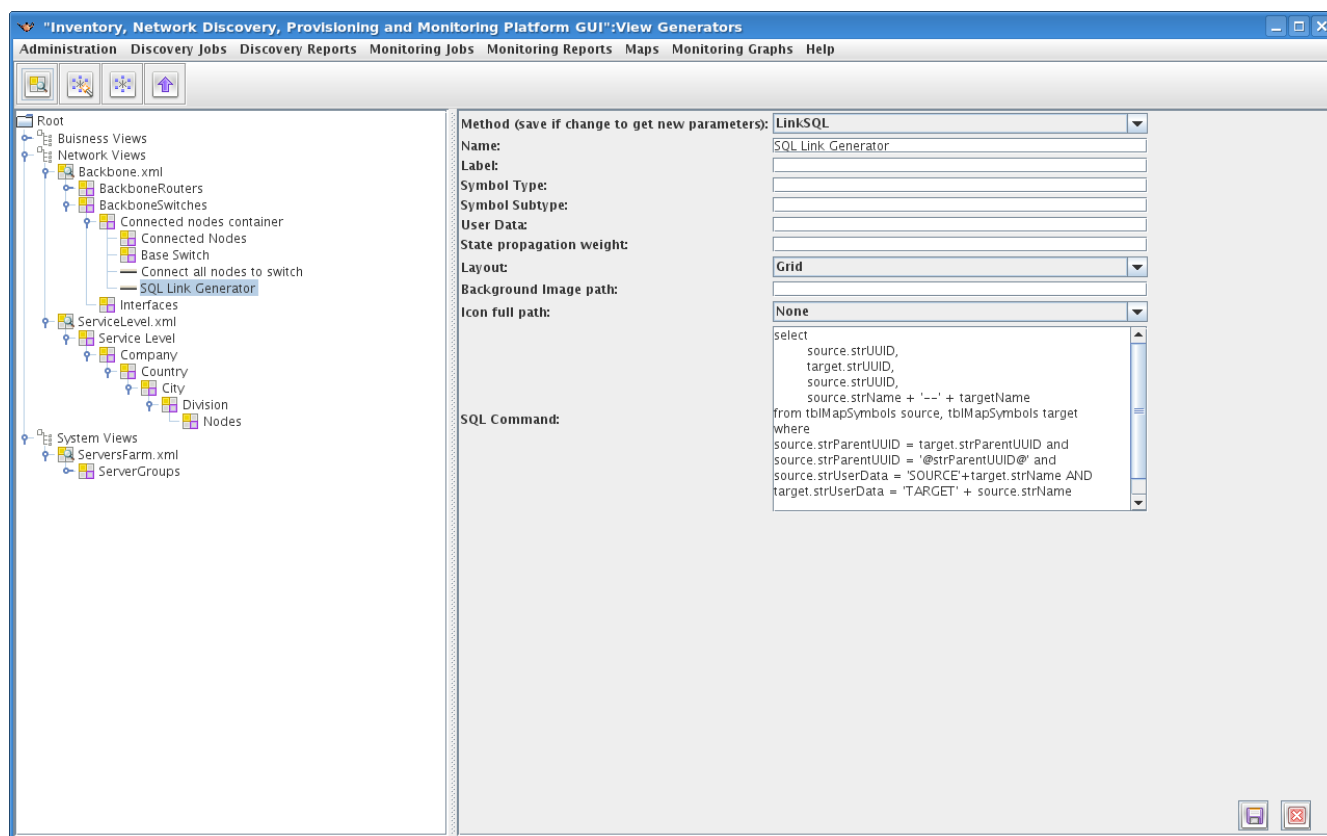
The name of the “from” symbols generator.

### ***To Symbols Generator name***

The name of the “to” symbols generator. It can match the one given in the “from” entry.

## 'Link SQL' Links Generator

The 'Link SQL' links generator connects symbols in a given sub-map based on a given SQL command.



Specific fields:

### **SQL command**

The sql command that defines the relations between the symbols that need to be linked by this links generator.

The SQL command should select at least 4 fields in the following order.

- StrSourceSymbolUUID – source symbol UUID
- strTargetSymbolUUID - target symbol UUID
- strColorSource – colour source of the link. The symbol which determines the color of the link (by default it is the source symbol).
- StrLabel – The label of the link

# Color Propagating

---

The mapping feature supports color propagation. Color propagation means that a child symbol can propagate its colors to its parent symbol and so on up to the view symbol.

The product supports a weighted propagation of colors where different symbols can have different influence on the parent symbol state and as such its symbol

## How do we map colors to states

The product supports a mapping between colors and states, and a mapping between a severity value to a state.

In the root directory of the view generators: [INSTALLDIR]/conf/viewgenerators there is a file named: "StateColorDefaultMapping.map" which defines the specified relationship.

```
# State to color mapping table
# -----
# colomns: state, color, low value , high value(non inclusive)
# the plus sign means > i.e 0+ means >0
Normal,green,0,0
Warning,yellow,0+,0.5
Severe,red,0.5+,1
```

The format of the file is:

# - marks a remark line

An entry line contains the following fields:

- State Name
- State Color
- Low Value of the state
- High Value of the state

When propagating a state from child to a parent, the state is translated in to the low value of the state, and then it is multiplied by the weight of the symbol, as specified in the symbol's definition.

The color of the parent is determined by the sum of all weighted values of its children.

---

**i** If a symbol is set a value directly then the propagation for this symbol is ignored.

---

## How to set the state of a symbol

By default if no state is given for a symbol, than its state is either “no state” or is taken from the states of its children.

There are a few methods to set a state for a symbol.

### Set an initial state for a symbol

In the symbols generators, the user can specify an initial state for a symbol.

## Change symbol state in Mapping database

The user can access directly the 'tblMapSymbols' table in the product, and change the symbol state directly using simple SQL commands.

# Final Page

---

## More Information

More information about Jilroy Software, Map Generator 3.0, 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 The Map Generator 3.0, 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.