

1. Reports introduction

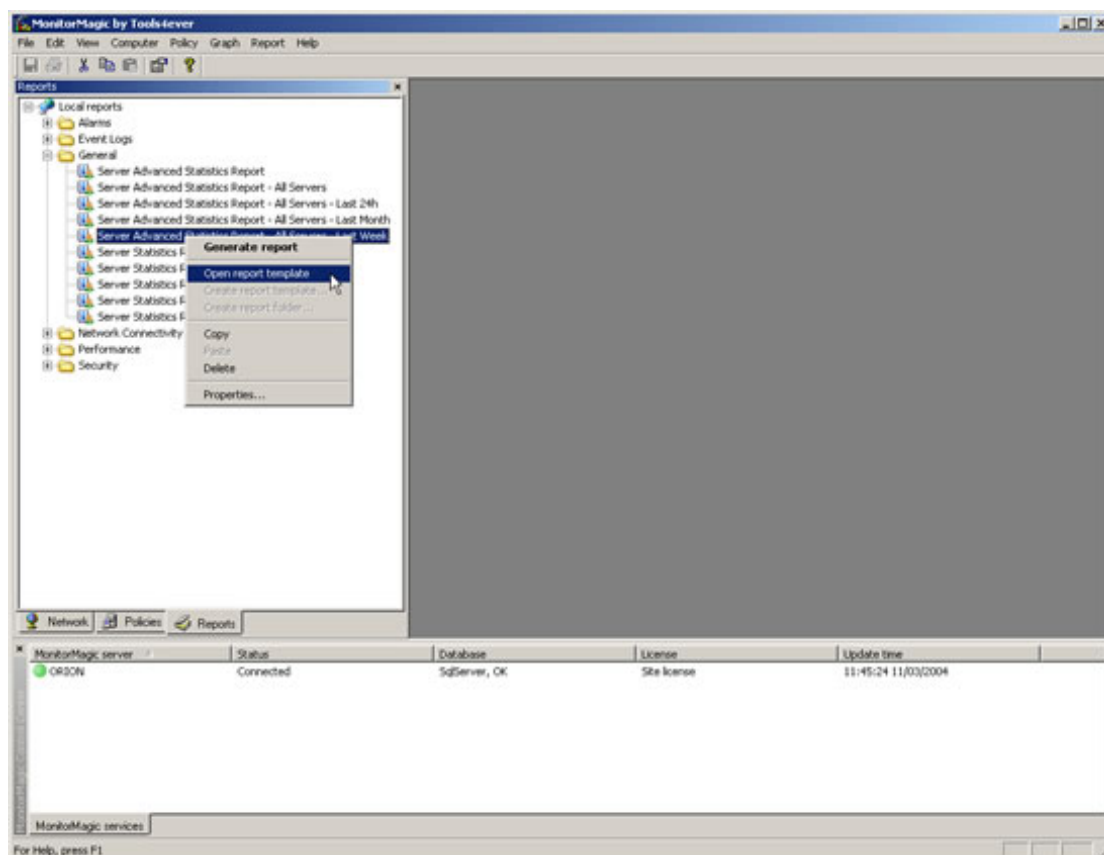
MonitorMagic features a built-in report generator, template builder and distribution scheduler. The top reasons to use these features instead of a 3rd party report engine such as Crystal Reports® are:

- **Dynamic reports.** The contents of the reports are automatically linked to the resources you are monitoring. Whether you monitor 5 or 50 servers, the reports do not require any configuration.
- **Iteration.** Report contents can be iterated over a list, for instance servers or even date ranges. The source of these iterations can be a fixed list, a SQL query or user prompted data. This feature enables reports which automatically display detailed information grouped per server, regardless of the number of servers.
- **Report e-mail distribution.** Reports can be automatically generated and distributed at a scheduled recurring date and time. This is for instance useful when you use the built-in weekly reports and set the distribution to Monday morning 8:30.

Note: when creating reports or customizing the pre-configured reports, basic knowledge of the Transact-SQL query language is required. Queries for each type of monitored resource are supplied in the reports; Tools4ever recommends using these queries as a starting point and modify them to suit your needs. This document is meant as an extension to the **MonitorMagic Implementation Guide** and certain parts assume that you have read and performed all operations in this document.

2. Working with reports

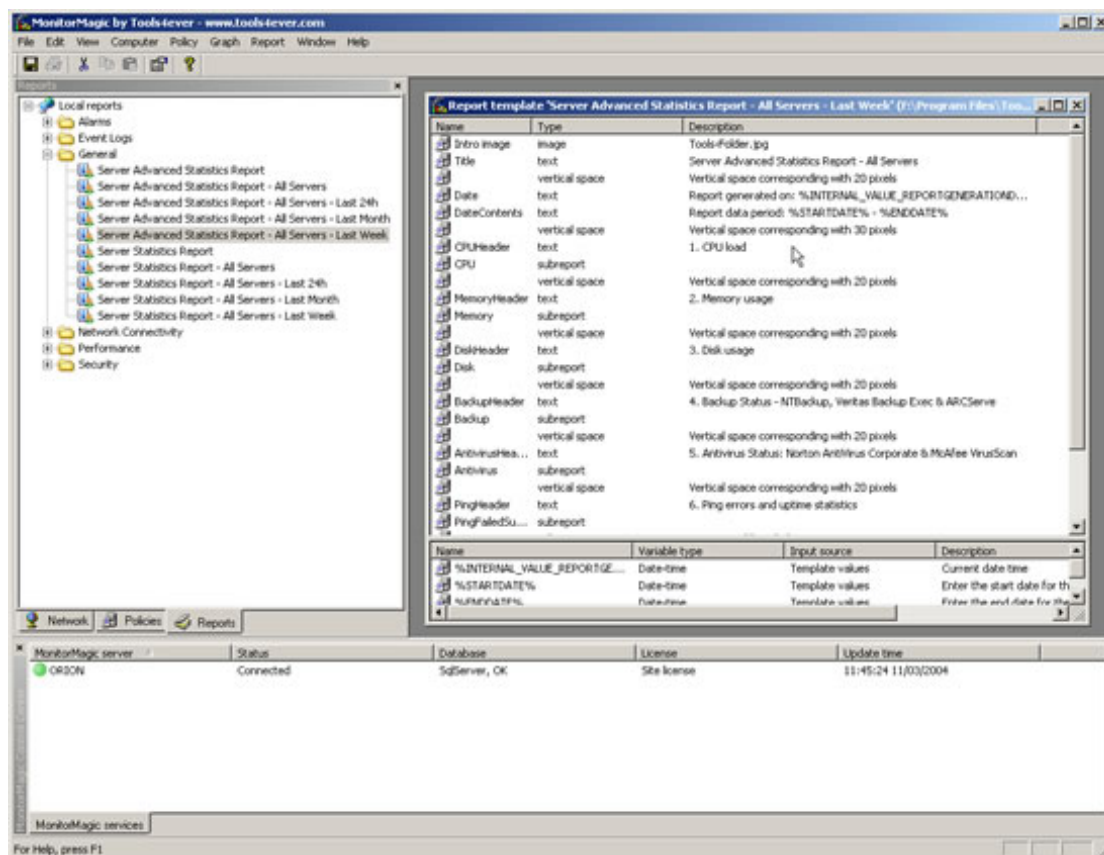
To start working with reports, switch to the "Reports" tab in the tree control on the left-hand side of the MonitorMagic application window. Select a report, in this example the "Server Advanced Statistics – All Servers – Last Week" is used, right-click and select "Open report template".



When the report template opens, you will see a new window containing a list of objects which represents the report template. Notice that the type of resource is listed in the 2nd column, and can be one of the following:

- **Text:** will display static text on the report
- **Table:** contains information retrieved by an SQL query
- **Subreport:** can contain any other element in a sub report
- **Graphic element:** will display a static image on the report. The **vertical space** is not actually a type of object, but rather a subset of the graphic element range.

When analyzing this table of objects, notice that the report starts with an introduction image, followed by a title. After this, we have a vertical space of 20 pixels, followed by two date objects. As you can see, both date objects use variables in their description, for instance **%STARTDATE%**. These variables are listed in the lower pane of this window in a separate table. Variables have two types, **user input** and **query**. Using these variables will be discussed in detail later on.

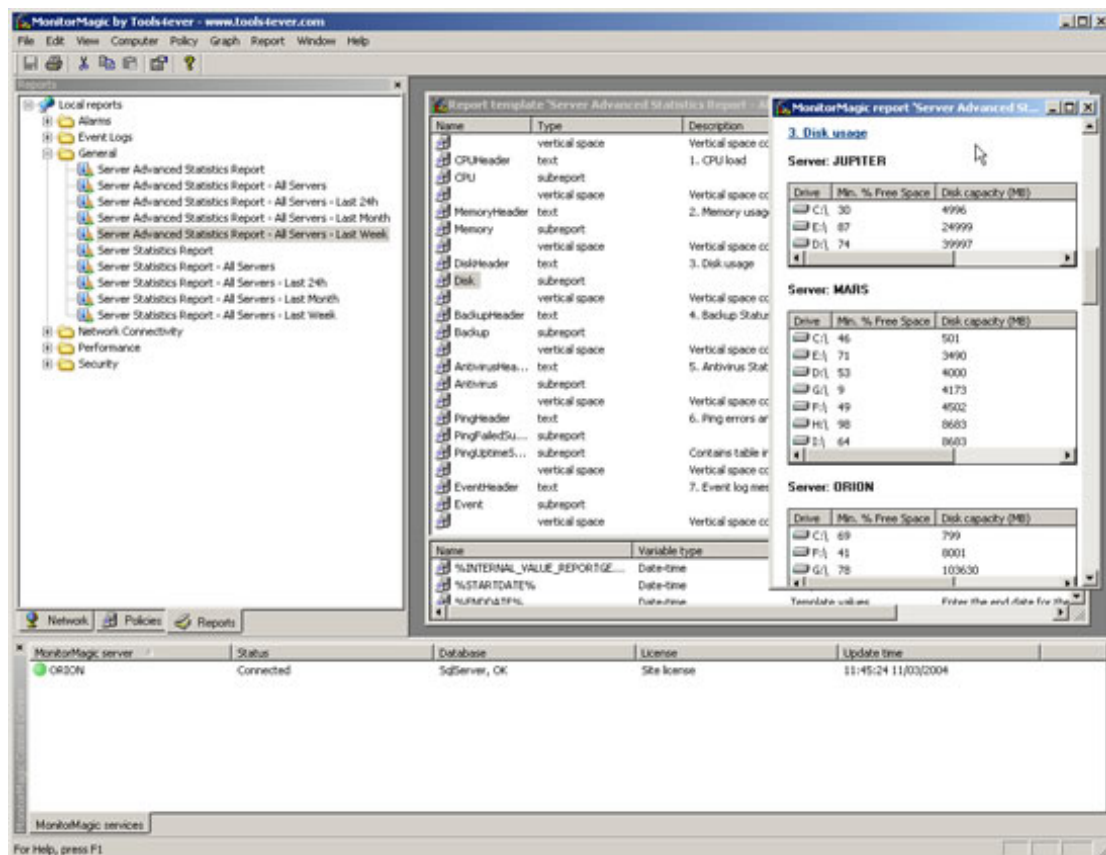


3. Understanding report objects

When you generate the report that we are using in this example by using right-click and select "Generate", or just double-click on the report template, arrange the windows so that they roughly match the below screenshot. To understand how the report is generated, let's compare the template objects with the final result, as shown in the screenshot below.

As you can see, the final result contains a section number 3, which contains disk usage information. This same section is represented as 2 objects in the report template, *DiskHeader* (a text object) and *Disk* (a subreport). The first object, when generated, will display the text "3. Disk usage". The tables you see in the generated report are the result of a subreport.

When you browse through the generated report and the report template objects, you can easily spot the similarities and make the link between the two.



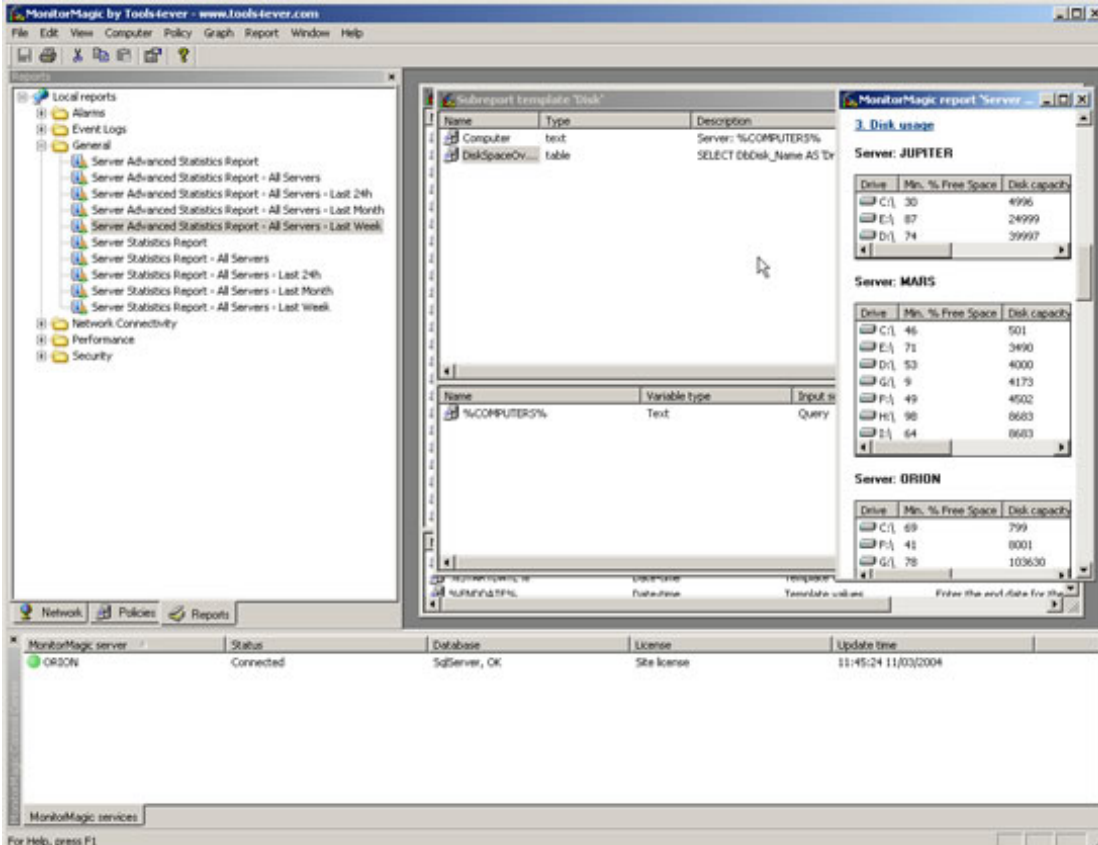
4. Understanding subreports

As we have seen in the previous chapter, the disk usage information in the report is represented by a text object and a subreport. Double-click on the subreport to view its contents similar to the screenshot below. Notice that a subreport is nothing more than a nested report in another report, so it can contain its own objects and variables of all different types as discussed earlier.

The usage scenarios for subreports are organization and iteration. To get a clean report template, divided into re-usable parts, subreports are great because you can easily copy/paste them into other reports while retaining all content. The iteration is what the subreport in our example is used for.

The subreport as you can see contains 3 objects, a text object which displays the computer name, a table which gets its data from the database connected to a MonitorMagic service, and a variable named %COMPUTERS%. When comparing these objects to what we see in the generated report, we notice that the two objects, the text and table, appear multiple times. This is because this subreport uses **iteration**. The power of iteration is to create dynamic reports based on the content of the database. For this report, we want to display disk space usage per server so we would have to get a list of all servers and then execute a certain SQL query for each server. This is exactly what this subreport does. *The iterator is the %COMPUTERS% variable.* When you double-click on this variable object, you will notice that there is a SQL query behind this. This query gets the servers that have disk data associated with them from the database and stores the value(s) in the variable %COMPUTERS%. When you now use this variable in the *DiskSpaceOverview* object, which is the table object representing the actual disk usage information, MonitorMagic automatically recognizes that the %COMPUTERS% is not a singular value, but in fact can contain multiple values (server names). MonitorMagic will then generate all objects inside the subreport for each value inside %COMPUTERS%.

The result, as you can see, is that both the text and table object are being generated for each server (JUPITER, MARS and ORION). The real power of iteration is that you no longer have to configure reporting for each server and that your report doesn't require input, which means you can use it for automatic distribution.



The screenshot shows the MonitorMagic software interface. The main window is titled "Subreport template 'Disk'". It contains a table with the following data:

Name	Type	Description
Computer	text	Server: %COMPUTERS%
DiskSpaceOverview	table	SELECT @@Disk_Name AS Dr
%COMPUTERS%	Text	Query

On the right side, there is a preview window titled "MonitorMagic report 'Server...'" showing the generated report for three servers: JUPITER, MARS, and ORION. Each server's report includes a table of disk usage with columns 'Drive', 'Min. % Free Space', and 'Disk capacity'.

Server: JUPITER

Drive	Min. % Free Space	Disk capacity
C:\	30	4996
E:\	87	24999
D:\	74	39997

Server: MARS

Drive	Min. % Free Space	Disk capacity
C:\	46	501
E:\	71	3490
D:\	53	4000
G:\	9	4173
F:\	49	4502
H:\	98	8683
I:\	64	8683

Server: ORION

Drive	Min. % Free Space	Disk capacity
C:\	69	799
F:\	41	8001
G:\	78	103630

At the bottom of the interface, there is a status bar showing the MonitorMagic server status for ORION, which is connected. The database is SqlServer, OK, and the license is Site license. The update time is 11:45:24 11/03/2004.

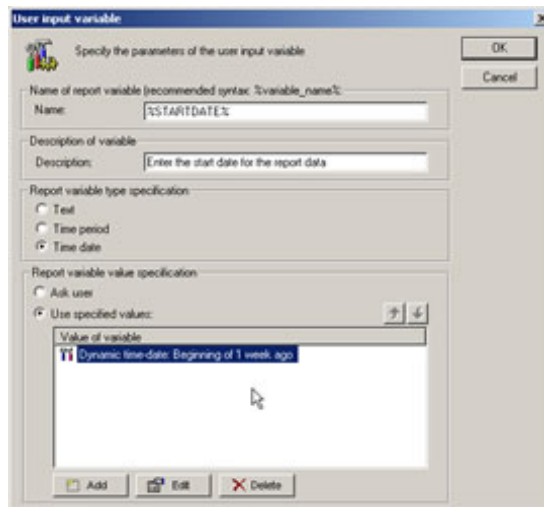
5. More on dynamic content

When generating the report, notice the lines of text just below the title. They contain date values which are dynamically generated upon generation. As we are working with the "**Server Advanced Statistics – All Servers – Last Week**", we only want to see information for last week. In the previous chapter we have seen how to get information from all servers, now we will discuss how to restrict the information to use the last week date range.

When you close the generated report and look to the variable objects listed in the report template, which are placed at the bottom of the template window in a separate pane, notice that 3 variables are listed: %INTERNAL_VALUE_REPORTGENERATIONDATE%, %STARTDATE% and %ENDDATE%. Double-click on the %STARTDATE% variable and notice that the content of this variable is being generated automatically by MonitorMagic. It will be assigned the date "**Beginning of 1 week ago**", which is convenient for our report. Similarly, the %ENDDATE% variable will contain the value "**Beginning of this week**". The benefit of having these variables defined globally is that you can use them in all subreports and SQL queries, and also that you can clearly present this date range at the beginning of the report.

The result is that the SQL queries both use the iterated variable %COMPUTERS% and the globally defined variables %STARTDATE% and %ENDDATE% to get the information from all servers using a specified date range.

If you need to modify the date ranges for this report, all you have to do is modify the value of the %STARTDATE% and %ENDDATE% variables. You can then generate this report without having to specify any additional input.



The final result will look similar to the screenshot below.

