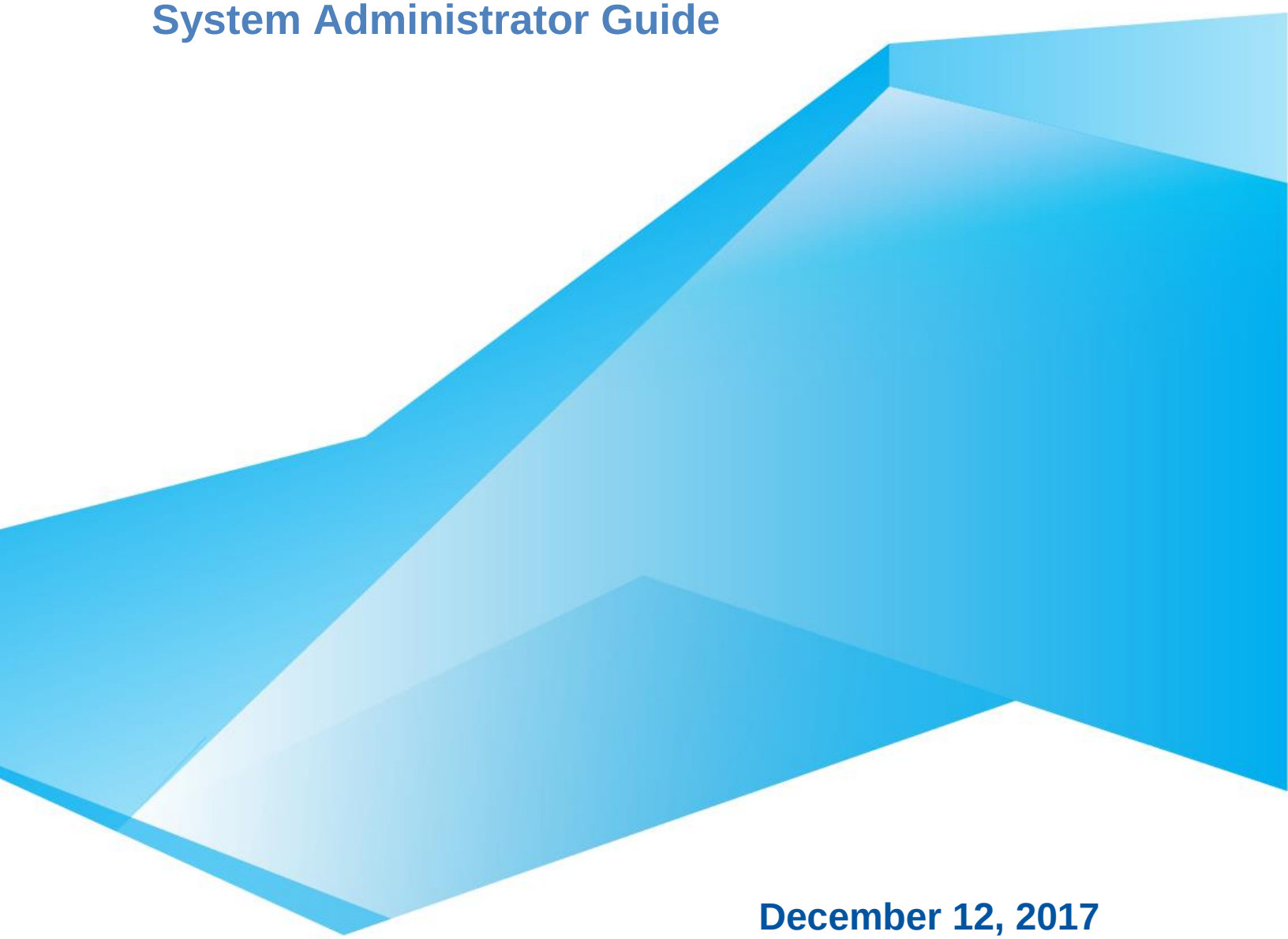
Deltek Know more. Do more.™

Deltek Maconomy® 2.3 GA

System Administrator Guide



December 12, 2017

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published December 2017.

Contents

Overview 1
Audience 1
System Description 2
Overview 2
Bandwidth Requirements Overview 4
Workspace Client Bandwidth 4
Portal and Java Client Bandwidth 7
Management Summary 7
Bandwidth Test 9
Detailed Network Test Results 14
Scenario Printouts 17
Hardware Setup 17
Integrations 19
Installation and Upgrade Overview 20
Standard Maconomy System 21
Release Documentation 21
Application Server 21
Legacy Server 21
Server 2.X 22
Proxy with Server 2.X 23
Intermediary Proxy 24
TCP Reverse Proxy – Example 28
Setting Up Proxy SSL on Server 2.X 31
Web Daemons 32
BPM 33
Clients 34
Analyzer Overview 36
Touch 40
Maconomy iAccess 40
Portal 40
Java Client 41
Update Sites 41
Solutions 43
Extensions 44
Configuration and Maintenance 45

Database Server 45
Application Server 46
Coupling Service 80
Background Task System Nature Setup 92
Advanced Logging 95
Silent Sign-In Overview 102
Security Overview 110
Single Sign-On 121
Single Sign On with Kerberos 126
Setup 130
Client Setup 130
Server Setup 133
Active Directory Setup 137
Example 137
SSO on the Macintosh 139
Using a Custom Verifier 140
Font Administration 141
Kona Integration 154
People Planner 157
General Server Maintenance 157
Performance Tuning, Debugging, and Logging 179
Java Client Debug Options 179
Portal Performance 186
Performance Monitor 196
Automation and Customization 219
Workflow Spider 219
MCron Overview 231
Getting Started with Localization 242
How to Update a Translation File 243
Tools and Resources 251
How to… 252
Clients 256
Maconomy Extender 261
RGL Report Designer Overview 261
Access Control View Templates 262
MDoc 267
How to Generate MDoc 267

How to Use MDoc 267
Standard Import Programs Overview 268

Overview

The System Administrator Guide defines the overall principles for installation and operation of a Maconomy installation. This guide is updated as new features are added or removed, but some functionality or details may be documented in separate guides.

Audience

The intended audience for this guide is:

- Maconomy System Administrators
- Database Administrators
- Technical Consultants
- Within Maconomy and Maconomy partners and customers

In addition, this guide may be beneficial for users who want to understand the components of Maconomy in more detail.

The first few sections are written as informative sections, whereas the remaining sections are expected to be used as a reference manual when specific actions are to be performed.

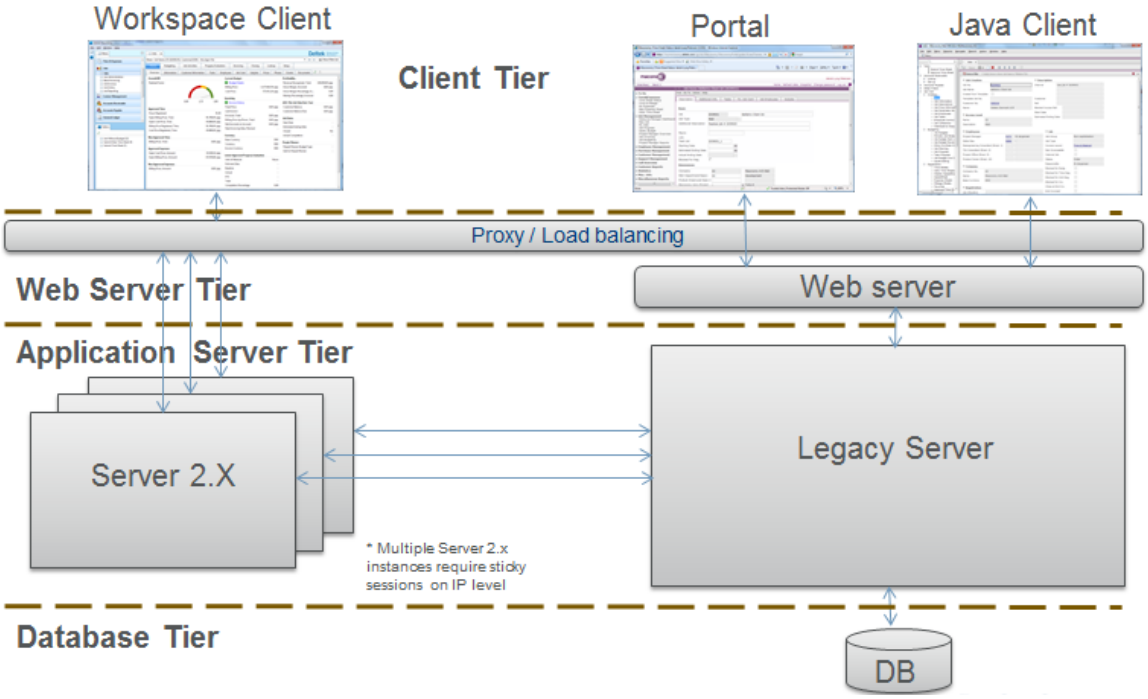
Notes on Forthcoming Technology Features

- **Scalable Server** — Scalable server will enable you to set up multiple Maconomy application servers accessing the same database to increase performance. (553561)
- **Maconomy Hardware Requirements Tool** — The Maconomy Hardware Requirements Tool will be a dynamic tool that allows you to input specific parameters based on your exact setup, and then generate custom recommendations. (552031)

If you need assistance with installing a scalable server or determining hardware requirements prior to release of these features, contact Chris Challenor, PDM (Maconomy Technology Product Manager).

System Description

A Maconomy system is a network-based, multi-tier solution that works in a network-based, distributed environment. The following figure shows Maconomy 2.X, including the services added in 2.X, and it shows a typical setup with more Server 2.Xs for redundancy and load sharing.



Deltek has from the initial development of Maconomy focused on following open standards and becoming independent from hardware and system software components. Each Maconomy version is certified on more hardware and software platforms. For details, see the release notes for the specific versions.

The Maconomy solution is a standard solution that is highly configurable to deliver exactly the functionality that is requested by the customer. The Maconomy solution can be used as both enterprise solutions and as cloud-based offerings; Deltek offers a cloud-based solution based on Maconomy, called “Deltek First Maconomy Essential” (DFME).

Maconomy solutions are very often integrated with other Deltek offerings, such as Deltek Touch or Deltek People Planner, or integrated with third-party systems, such as data warehouses. Deltek’s Maconomy team has incorporated integrations with a set of other Deltek offerings and created standard interfaces for building integrations with third-party products.

The physical layout of servers and infrastructure is very customer-specific, starting with everything on one server, to scaling out to multiple servers, one for each Maconomy component, or even several equal servers to handle load sharing (only possible for some parts of Maconomy). This guide describes how to set up and configure Maconomy.

Overview

As illustrated in the preceding figure, a Maconomy system contains several different server components and different client interfaces.

The traditional client interfaces—Java Client and Portal—have been part of the Maconomy delivery in many previous versions, whereas the Workspace Client was introduced in Maconomy

2.X. The traditional client interfaces are still maintained, but they will be deprecated and replaced by the Workspace Client and a future HTML client.

All client interfaces communicate with a server complex that in the smallest installation all reside on one computer; a larger installation is scaled out on multiple computers and in more instances.

This guide focuses on the software and discusses how to install, tune, customize, and operate the solution on the selected hardware setup, whereas the physical hardware configuration is designed and selected as part of the customer implementation project.

Standard Components

When planning a Maconomy system, it is important to decide what to install and how to scale and configure the hardware to meet the solution requirements. There are several components and add-ons that may be beneficial for some users.

A standard installation has a set of technical components: database (SQLserver or Oracle), Maconomy legacy server, Maconomy Server 2.X, web server, and one or more client interfaces (Workspace Client, Java client, or Portal), as well as the Maconomy Application package.

In addition to these standard components, a customer installation may have a solution installed—either CPA or PSO—and, if beneficial, one or more of the standard Industry Accelerators.

The combination of selected standard components matches a certain percentage of the customer needs, and can be further customized to fully comply with requirements.

Solutions

Two solutions: CPA and PSO

Industry Accelerators

Deltek delivers and supports some product Industry Accelerators, which are certified to each Maconomy release. Some IAs may be added, and some may be removed in newer Maconomy releases. Some examples of IAs are:

- Absence Management
- Expense Justification

IAs are installed separately on already existing Maconomy installations. Installation is managed using Mconfig, and some manual configuration may be required to integrate with customizations made on the customer system, for example.

Customizations

Customer systems may be customized to make the system comply with customer-specific or domain-specific requirements.

Maconomy can to a large degree be customized by configuring the system using Mconfig or using the client interfaces with Administrator rights.

Some customization may require that special configuration files, layouts, or even software enhancements be added to the standard installation. This type of customization is performed using the dedicated development environment, Maconomy Extender. For more information, see the Maconomy Extender manual.

Bandwidth Requirements Overview

This document describes bandwidth testing for Maconomy 2.0+.

This document describes recommendations and limitations that need to be considered when defining a Maconomy solution for customers. Several tests and measurements have been made and taken into consideration when outlining the following recommendations. The results for the Workspace client cannot be directly compared with the results for the Java client and Portal because the methods used are very different.

This document or subsections of it cannot be part of any Maconomy contract because assumptions can be made that cannot be directly transferred to a production system.

Workspace Client Bandwidth

Many Maconomy customers are global companies with offices in several countries. They must consider bandwidth and latency between these offices and their central Maconomy installation.

This section provides information about the bandwidth required to handle the connections between the Workspace Client (WSC) and the Coupling Service (CS), as well as estimates of the consequences that network latency has on the perceived performance of the WSC.

The theoretical performance impact of network latency is calculated for a number of different latency figures and combined request/response data package sizes, covering the conditions that can be expected from an unspecified remote office location.

Network Calculations

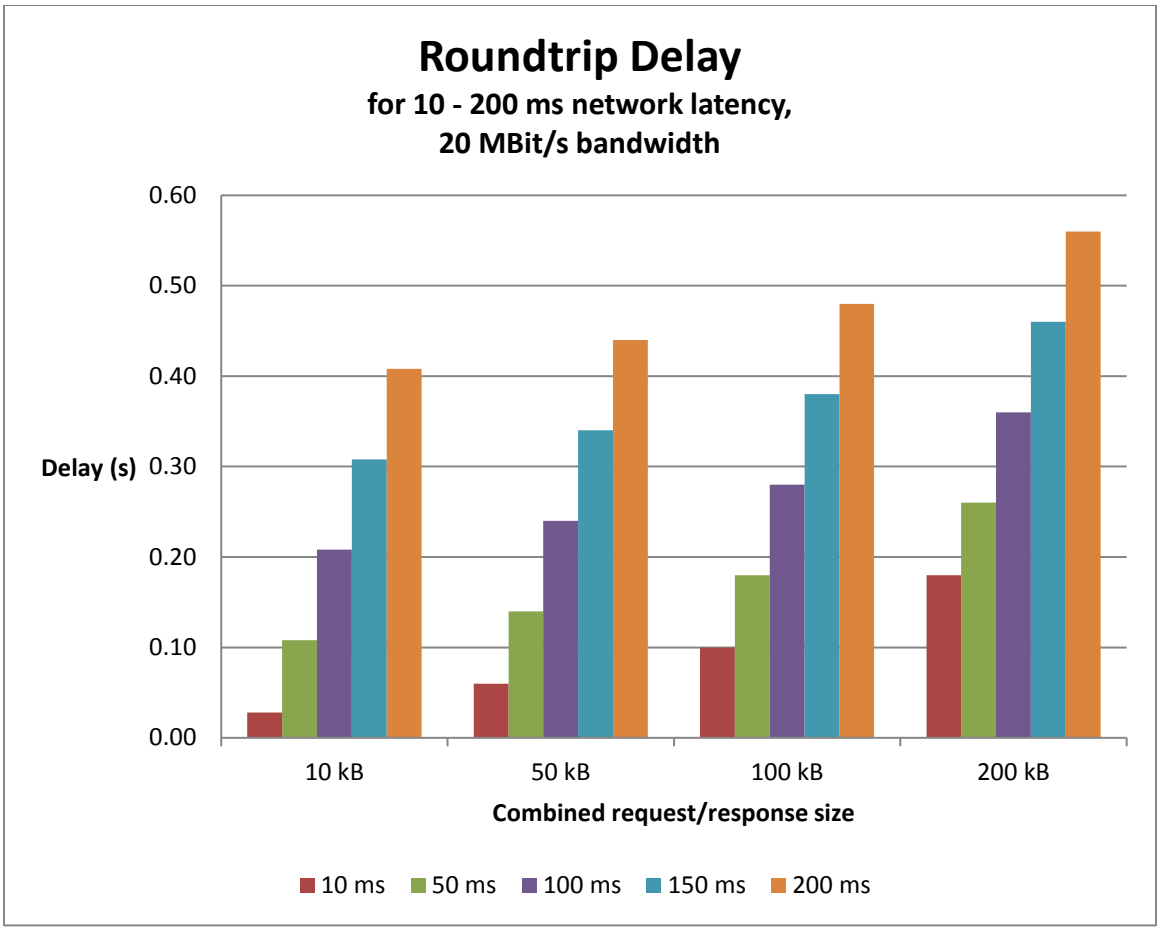
Using a few simple formulas it is possible to calculate the roundtrip delay (RTD) for a request/response pair with a given message size when passing data across a network with a known latency and bandwidth. The RTD is a calculation of the average time it takes to send a request and receive the response, assuming that the data processing at the remote end takes zero time.

Roundtrip Delay

For calculating a RTD use the following formula:

RTD = (2 * Network Latency) + (Message Size / Network Bandwidth)

The following figure shows the calculated RTD incurred on a 20 Mbit/s network for different latencies and message sizes in a single roundtrip from a WSC to the CS. This shows that for small latencies the RTD is also small, and it is decided by the message size. As the latency grows, so do the RTD values and the message size (and therefore network bandwidth) becomes less relevant:



Logical Operation Delay

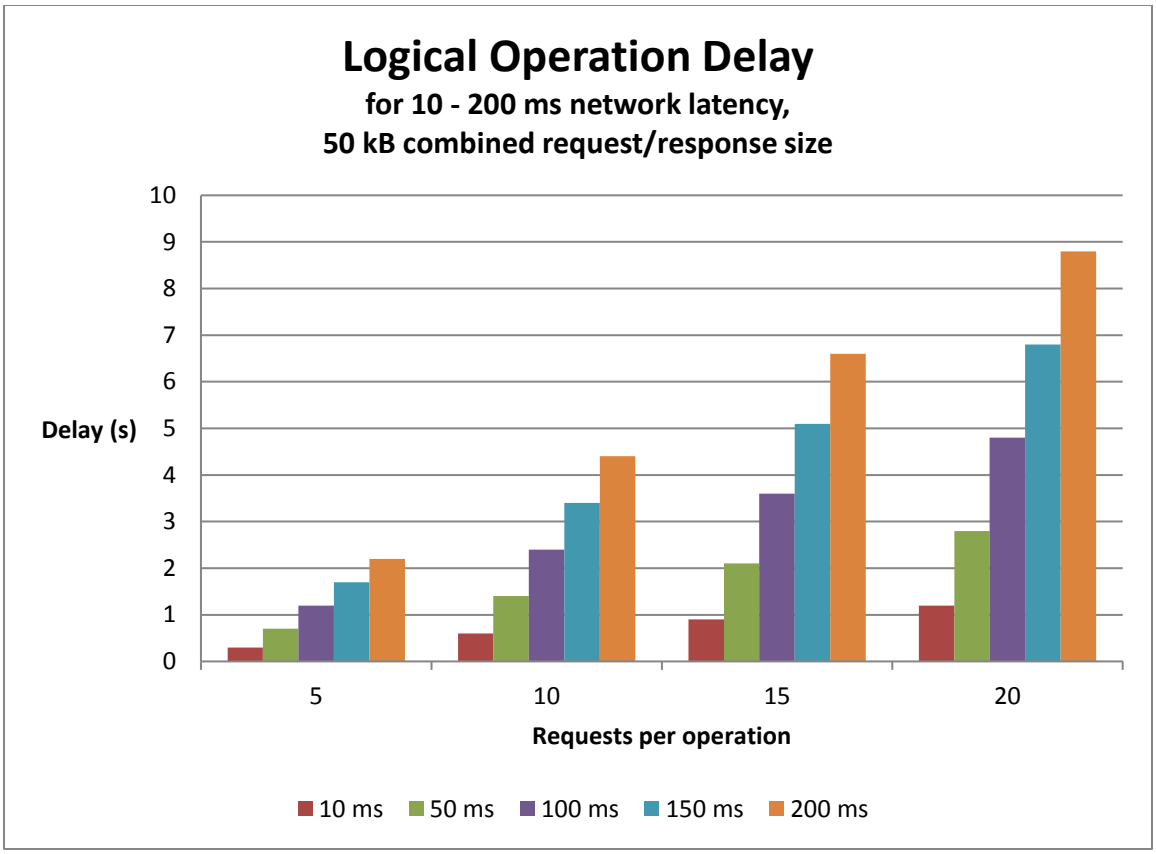
The next figure shows the “logical operation delay” for a complete workflow in the Workspace Client, as a function of the number of request/response roundtrips between the WSC and the CS.

An example workflow could be “create and fill out a time sheet line” or “view and approve 10 project entries”. The logical operation delay is then the total time spent waiting on the network while executing the workflow.

For these calculations, assume an average request/response message size of 50 kB. Assume also that the typical workflow consists of 5 – 20 server requests.

Once again it is apparent that the network latency is a major factor; workflows that are executed with a total delay of around 1 second on a low-latency network suffer from a total delay of 5-8 seconds on a high-latency network.

Depending on the operations being performed in the workflow this could translate to an increase in the total workflow execution time of up to 40% for small operations, as illustrated by the following figure.



Network Bandwidth

The following table focuses on the total volume of network traffic across the network as a function of the number of active users on the system. This table provides the calculated total network traffic (in Mbit/s) for 100 – 1000 users and 5 different message sizes.

These figures can be used to get an estimate for the network bandwidth that should be reserved to handle WSC requests to handle peak load from a specific number of users without experiencing network congestion. The 50 kB column is highlighted because it is considered a reasonable conservative estimate for the typical average WSC request/response message size:

Number of users \ Combined request/response size	5 kB	10 kB	50 kB	100 kB	200 kB
100	1	1	**4**	8	16
200	1	2	**8**	16	32
300	2	3	**12**	24	48
400	2	4	**16**	32	64
500	3	5	**20**	40	80

	600	3	6	**24**	48	96
	700	4	7	**28**	56	112
	800	4	8	**32**	64	128
	900	5	9	**36**	72	144
	1000	5	10	**40**	80	160

Conclusion

Based on the preceding calculation it is now possible to calculate the required bandwidths and the influence of latency.

For example, if an installation has 250 users located in remote locations, it requires a bandwidth of approximately 10 Mbit/s, whereas if there are fewer than 100 users in a remote location, it requires approximately 4 Mbit/s.

The roundtrip delay (RTD) for a request/response roundtrip between a remote location and the central Maconomy installation can be calculated as the following example: if the average message size is 50kb, and the latency is 150ms, the RTD is expected to be approximately 300 – 400ms. The Logical Operation Delay for typical WSC workflow scenarios is expected to be approximately 5 – 8 seconds, or up to 40% for small operations.

Factors that could invalidate these results include:

- Changes in network bandwidth and/or latency, either due to changes in the network topology or because other services on the same network are causing congestion.
- A usage pattern during peak hours that differs significantly from the assumed average of 0.1 requests per user per second.
- An average message size significantly greater than the estimated 50 kB per request/response pair. Such an increase in message size could be the result of things such as disabling compression on the WSC – CS network connection, or by users performing common tasks in workspaces with unnecessarily many panes open.

Portal and Java Client Bandwidth

This section describes bandwidth testing performed by Deltek. This assumes that the reader is familiar with Maconomy technical terms and architecture.

This section describes the bandwidth usages of a default Project Manager in the Maconomy Consulting Solution (MCS). These tests have been performed using a suite of bandwidth measurement tools such as LoadRunner, CommView, and Ethereal.

Management Summary

Deltek has conducted a network load using the most frequently used data entry points—that is, time sheet and expense sheet entry. Two profiles were selected for this test:

1. A standard Project Manager in a Consulting Solution. This role was chosen because it reflects the most common user setup in the installed customer base.

2. A Time & Expense user with a minimum user role. This role was defined specifically for entering time and expense sheets and reflects the minimum bandwidth uses possible for these scenarios.

All statistics are measured with cache enabled on the browser and no first-time executions.

Network Load

The user profiles used in these tests are a standard Maconomy Project Manager user (PM-role) with HR access and a Time & Expense user (TE-Role). This TE-role consumes less bandwidth because time and expense users do not need Java applets.

Three scenarios were designed for this test:

1. A time sheet that contains a copy from the previous week (nine lines), where three lines were changed. A line was added to the end, and the time sheet was submitted.
2. An expense sheet with ten lines.
3. A search for a job number (CTRL+G search).

These tests were performed using the Maconomy Portal with and without web server compression and a Java client.

The following results were retrieved. The figures show the number of bytes transferred over the network per operation as described previously.

Project Manager Role **PM-Role**	Time Sheets	Expense Sheets	Search
	Total Bytes	Total Bytes	Total Bytes
Portal	339,915	485,871	178,915
Portal, Compressed	222,655	299,755	23,760
Java Client	292,357	316, 614	N/A

TE Role	Time Sheets	Expense Sheets	Search
	Total Bytes	Total Bytes	Total Bytes
Portal	191,382	335,383	178,710
Portal, Compressed	87,817	164,961	23,945
Java Client	284,140	307,857	N/A

Time sheet entry using the TE role achieved the lowest bandwidth measurements when using Apache with compression enabled. This combination uses only 88 Kb. Without compression, the standard Portal uses 191 Kb. The PM role requires 223 Kb with compression and up to 340 Kb without.

The bandwidth requirements between the database server and the web server are in the range from 340 Kb for the TE role and up to 575 Kb for the PM role. The user role has a huge impact on bandwidth usage. A standard PM role uses Java dialogs in the Portal, whereas a TE role does not. Using Java dialogs increases bandwidth usage from 191 Kb to 340 Kb for time sheet entry,

which is an increase of about 77%. It is therefore recommended that you consider low-bandwidth roles for remote users on limited-bandwidth lines.

For expense sheet entry, the TE role again shows the most effective bandwidth usage with 165 Kb using compression and 335 Kb without compression. The PM role uses 300 Kb with compression and 486 Kb without compression.

Enabling compression on the web server adds an overhead of approximately 5% CPU power. Despite this fact, based on the preceding results Deltek recommends centralized web servers with compression enabled if bandwidth is an issue.

Bandwidth Test

This bandwidth test was performed in September and October, 2005. Maconomy version 9.0 was used with the latest patch level. All servers were reinstalled with Windows 2003 Service Pack 1. For a detailed description of the hardware setup, see “Hardware Setup.”

Background and Test Objectives

The objective of this test was to test how great a load Maconomy transactions inflict on the network when doing data entry.

The purpose was to establish a common base for estimating network load in a new Maconomy installation based on expected transaction volumes and user behavior.

The greatest simultaneous data entry process in Maconomy is time and expense sheet entry, meaning that a large number of users often need to enter their time and expense sheets at the same time. Therefore a user behavior for entering these sheets has been defined, and the following network bandwidth tests have been performed:

1. Time sheet entry using a Project Manager profile (PM role).
2. Time sheet entry using a pure Time & Expense sheet profile (TE role).
3. Search for a job number in the Portal.

The Project Manager user profile has HR access, whereas the Time & Expense sheet user only has access to entering time and expenses. This profile consumes less bandwidth because Time & Expense users do not need Java applets.

All tests were performed using the Maconomy interfaces:

1. Maconomy Portal
2. Maconomy client for the Java™ platform (“Java client”)

This test measures the network load from one user entering one time sheet. This is done by entering data and measuring the network traffic between the following units:

- The Client and the web server
- The web server and the application/database server

All results were measured three to four times to ensure the quality of the figures used. The maximum deviance between the different measurements was 4%.

For a more detailed description of the test results see “Detailed Network Test Results.”

Time Sheet Entry

The scenario for entering a time sheet in Maconomy was defined as follows:

1. Open the login page / Launch the client.

2. Log in to Maconomy.
3. Open the Time Sheets window (Java and back office clients only).
4. Create a time sheet.
5. Copy nine lines from the previous week.
6. Update three lines with a quantity of hours.
7. Add a line by entering the job number, task, and quantity of hours.
8. Submit the time sheet.
9. Exit Maconomy.

The preceding test does not include any searching, because the network load generated from searches depends entirely on user behavior (such as parameter and search criteria selection) and can therefore not be tested in general.

Test Results, PM Role

The test results are summarized in the following table. (See “Detailed network Test Results” for a more detailed description of the test results.)

Web Interfaces

	Bytes Sent to Web Server	Bytes Rec’d from Web Server	Total, Web Client	Bytes Sent to App Server	Bytes Rec’d from App Server	Total, App-Web
Portal	56,817	283,098	339,915	143,260	431,619	574,879
Portal, Compressed	54,484	169,171	222,655	138,166	426,090	564,256
Java Client	67,550	224,807	292,357	111,029	259,211	370,240

Bandwidth usage between the application and the web server is in the range of 370,240 bytes using the Java client to 574,879 bytes using the Portal with compression. The difference between the Portal with and without compression is 10,623 bytes, or less than 2%. As expected, the traffic between the application server and the web server is the same regardless of compression setting.

The total amounts of data sent between the different types of web-based clients are in the range of 222,655 and up to 339,915 bytes. Using the Portal with compression enabled is the most efficient interface.

Test Results, TE Role

The test results are summarized in the following table. (See “Detailed Network Test Results” for a more detailed description of the test results.)

Web Interfaces

	Bytes Sent to Web Server	Bytes Rec'd from Web Server	Total, Web Client	Bytes Sent to App Server	Bytes Rec'd from App Server	Total, App-Web
Portal	44,769	146,613	191,382	88,622	251,939	340,561
Portal, Compressed	42,354	45,463	87,817	91,690	258,079	349,769
Java Client	65,739	218,401	284,140	112,144	260,325	372,469

Bandwidth usage between the application and the web server is 372,469 bytes using the Java client and 340,561 bytes using the Maconomy Portal. Only 31,908 bytes or 9.3% separate the lowest and the highest levels of bandwidth use.

The lowest bandwidth measurement was obtained using Apache with compression enabled. This combination uses only 87,817 bytes. The traffic between the application server and the web server is 4 times higher than the amount of data sent to the client.

The total amount of data sent using the Java client is 284,140 bytes between the client and the web server, where the standard Portal uses 191,382 bytes.

Conclusion

Approximately 36% of the communication between the web server and the client can be eliminated by using an Apache web server with compression enabled. The Portal with compression enabled is the less-bandwidth-consuming interface with 222,655 bytes sent and received using the PM role and only 87,817 bytes using the TE role.

The amounts of data sent between the application server and the web server vary from 340,561 bytes to 574,879 bytes, both figures with compression disabled. The difference of 234,318 bytes can be attributed to the difference between the PM role and the TE role. The figures show that within the same role, the bandwidth is the same (within a 5% margin).

Please note that the bandwidth consumption between these servers is very dependent on M-Script functionality. During this test, the M-Script functionality was very basic, such as getting one line and showing it in the client. If the M-Script code is more complex, the bandwidth usage between the application and web servers increases dramatically.

Because of the number of bytes sent between the application and the web servers, the most efficient way to place these two servers is on the same local area network.

Expense Sheet Entry

The scenario for entering an expense sheet in Maconomy was defined as follows:

1. Open the login page / Launch the client.
2. Log in to Maconomy.
3. Open the Expense Sheets window (Java and back office clients only).
4. Create an expense sheet.
5. Add one line by entering data in the **Job No.**, **Task**, **Description**, **Voucher No**, and **Amount in GBP** fields.
6. Add nine lines like the one described in step 5.

7. Submit the expense sheet.
8. Exit Maconomy.

Test Results, PM Role

The test results are summarized in the following table. (See “Detailed Network Test Results” for a more detailed description of the test results.)

Web Interfaces

	Bytes Sent to Web Server	Bytes Rec’d from Web Server	Total, Web Client	Bytes Sent to App Server	Bytes Rec’d from App Server	Total, App-Web
Portal	104,491	381,380	485,871	237,094	530,062	767,156
Portal, Compressed	101,121	198,634	299,755	238,265	531,280	769,545
Java Client	100,511	216,103	316,614	155,152	273,034	428,186

Bandwidth usage between the application and the web servers is in the range of 428,186 bytes using the Java client and up to 769, 545 bytes using the Portal interface. Again, the same bandwidth is measured when the Portal interface is used, regardless of compression.

The lowest result is achieved when compression is enabled and the Portal is used to enter expense sheets. The number of bytes sent between the web server and the client was 299,755 bytes. Without compression, the same scenario used 485,871 byes, or 62% more bandwidth.

The Maconomy client for the Java platform is nearly as efficient as the Portal without compression, but it is important to mention that the functionality of the Java client is different from the Portal interface. To create an expense sheet, you must open the Expense Sheet window. When this is done the last expense sheet that was entered is displayed. In this case, the last entered expense sheet is exactly like the one used in this scenario, and this operation uses approximately 45,000 bytes.

Test Results, TE Role

The test results are summarized in the following table. (See “Detailed Network Test Results” for a more detailed description of the test results.)

Web Interfaces

	Bytes Sent to Web Server	Bytes Rec’d from Web Server	Total, Web Client	Bytes Sent to App Server	Bytes Rec’d from App Server	Total, App-Web
Portal	92,723	242,660	335,383	1887,142	358,945	546,087
Portal, Compressed	90,712	74,249	164,961	187,382	358,999	546,381

Java Client	98,952	208,905	307,857	158,715	268,187	426,902

Bandwidth usage between the application and the web servers is in the range of 426,902 bytes to 546,381 bytes. These values are still much larger than the amount of data sent from the web server to the client.

The figure for the Portal with compression is really low, with a total of 164.961 bytes. To enter the same expense sheet, twice the bandwidth is needed when another interface is used. All other interfaces lie in the range of 307,857 bytes and up to 363,114.

Conclusion

Entering expense sheets using the Portal with compression is the most efficient way to minimize bandwidth usage. The TE role uses 164,961 bytes, compared with 299,755 bytes for the PM role, or more than 80% less bandwidth for the exact same scenario. It is therefore very important to consider the design of the user roles for users with limited bandwidth.

Placing the web server and the application server on the same subnet with fast connections between them is essential for good performance. All web interfaces used less bandwidth between the client and the web server than the bandwidth between the web server and the application server.

Search Jobs

The scenario for searching job numbers in Maconomy was defined as follows:

1. Open the login page / Launch the client.
2. Log in to Maconomy.
3. Create an expense sheet.
4. Place the cursor in the **Job No.** field and press CTRL+G.
5. Press Enter in the search window and list the first 15 jobs.
6. Select line 12.
7. Return to the expense sheet.

Test Results, PM Role

Web Interface

	Bytes Sent to Web Server	Bytes Rec'd from Web Server	Total, Web-Browser
Portal, Compressed	14,268	164,647	178,915
Java Client	10,289	13,471	23,760

It is apparent that compression is very efficient in searches. The reason is that the search result set is sent back to the client in the largest possible package size, and that all data is in text format and therefor easy to compress effectively.

Test Results, TE Role

Web Interface

	Bytes Sent to Web Server	Bytes Rec'd from Web Server	Total, Web-Browser
Portal, Compressed	14,114	164,596	178,710
Java Client	10,351	13,594	23,945

Compression is very efficient when searching in the Portal, regardless of user role.

Conclusion

The TE Role and the PM Role use the same bandwidth in this scenario, which is obvious because the functionality is exactly the same. In environments where many searches are performed, Apache with compression is a very effective combination. If compression is disabled, 6.7 times more bandwidth is required.

Detailed Network Test Results

Time Sheet	PM Role Portal			TE Role Portal		
	Sent	Receive	Total	Sent	Receive	Total
Open Portal	3.983	6.556	10.539	4.189	6.616	10.805
Login	19.911	166.122	186.033	10.732	31.414	42.146
Open Time Sheet	6.576	77.373	83.949	6.352	77.349	83.701
Create Time Sheet	2.244	1.731	3.975	2.296	1.684	3.980
Copy 9 lines	2.081	2.805	4.886	2.079	2.412	4.491
Edit 3 lines	6.777	3.428	10.205	6.904	3.548	10.452
Add 1 line	4.768	2.605	7.373	4.426	2.355	6.781
Submit	2.684	10.701	13.385	2.678	10.665	13.343
Logout	7.793	11.777	19.570	5.113	10.570	15.683
	56.817	283.098	339.915	44.769	146.613	191.382

	Portal, Compressed			Portal, Compressed		
	Sent	Receive	Total	Sent	Receive	Total
Open Portal	3.963	4.568	8.531	3.983	4.570	8.553
Login	18.959	135.367	154.326	10.434	14.547	24.981
Open Time Sheet	5.064	11.533	16.597	4.902	11.375	16.277
Create Time Sheet	2.244	1.191	3.435	2.020	1.008	3.028
Copy 9 lines	2.135	1.258	3.393	2.135	1.253	3.388
Edit 3 lines	6.831	2.592	9.423	6.816	2.593	9.409
Add 1 line	4.765	2.024	6.789	4.752	2.083	6.835
Submit	2.460	1.854	4.314	2.457	1.903	4.360
Logout	8.063	7.784	15.847	4.855	6.131	10.986
	54.484	168.171	222.655	42.354	45.463	87.817

	Java Client			Java Client		
	Sent	Receive	Total	Sent	Receive	Total
Open Portal	16.162	30.535	46.697	14.411	26.894	41.305
Login	4.731	104.201	108.932	4.649	101.970	106.619

Open Time Sheet	5.205	62.569	67.774	5.960	62.220	68.180
Create Time Sheet	6.224	3.114	9.338	6.149	3.203	9.352
Copy 9 lines	6.006	6.492	12.498	5.816	6.327	12.143
Edit 3 lines	14.344	6.575	20.919	14.194	6.617	20.811
Add 1 line	8.607	3.500	12.107	8.446	3.574	12.020
Submit	3.239	6.610	9.849	3.265	6.568	9.833
Logout	3.032	1.211	4.243	2.849	1.028	3.877
	67.550	224.807	292.357	65.739	218.401	284.140

Expense Sheet	PM Role Portal			TE Role Portal		
	Send	Receive	Total	Send	Receive	Total
Open portal	3.988	6.616	10.604	3.988	6.556	10.544
Login	19.987	167.77	187.76	10.933	31.714	42.647
Open Expense			0			0
Create Expense	13.832	92.908	106.74	13.886	92.824	106.71
Add 1st line 3 lines	7.493	10.754	18.247	7.277	10.514	17.791
Add 9 lines	46.381	83.731	130.11	46.865	84.192	131.05
Submit	2.710	5.818	8.528	2.818	5.938	8.756
Logout	10.100	13.779	23.879	6.956	10.922	17.878
	104.49	381.38	485.87	92.723	242.66	335.38

	Portal, Compressed			Portal, Compressed		
	Send	Receive	Total	Send	Receive	Total
Open portal	4.233	4.986	9.219	4.096	4.693	8.789
Login	19.013	135.192	154.205	10.555	14.670	25.225
Open Expense						
Create Expense	12.719	20.164	32.883	12.055	19.477	31.532
Add 1st line 3 lines	7.331	3.836	11.167	7.223	3.716	10.939
Add 9 lines	45.029	23.388	68.417	46.960	23.804	70.764
Submit	2.697	1.514	4.211	2.697	1.513	4.210
Logout	10.099	9.554	19.653	7.126	6.376	13.502
	101.121	198.634	299.755	90.712	74.249	164.961

	Java Client			Java Client		
	Send	Receive	Total	Send	Receive	Total
Open portal	14.912	30.759	45.671	13.641	27.134	40.775
Login	4.759	104.323	109.082	4.189	101.790	105.979
Open Expense	4.000	42.368	46.368	3.976	41.036	45.012
Create Expense	8.326	2.910	11.236	8.262	2.919	11.181
Add 1st line	6.615	3.093	9.708	6.598	3.093	9.691
Add 9 lines	57.636	25.683	83.319	57.706	25.805	83.511
Submit	2.351	6.000	8.351	2.363	5.978	8.341
Logout	1.912	967	2.879	2.217	1.150	3.367
	100.511	216.103	316.614	98.952	208.905	307.857

Search (CTRL+G)	PM Role Portal			TE Role Portal		
	Sent	Receive	Total	Sent	Receive	Total
Open Search	6.912	146.458	153.370	6.704	146.347	153.051
Find jobs (15 rows)	2.699	11.410	14.109	2.645	11.410	14.055
Select row 12	4.657	6.779	11.436	4.765	6.839	11.604
	14.268	164.647	178.915	14.114	164.596	178.710

	Portal, Compressed			Portal, Compressed		
	Sent	Received	Total	Sent	Received	Total
Open Search	3.940	9.704	13.644	3.778	9.645	13.423
Find jobs (15 rows)	1.998	1.634	3.632	2.222	1.816	4.038
Select row 12	4.351	2.133	6.484	4.351	2.133	6.484
	10.289	13.471	23.760	10.351	13.594	23.945

Data Transmitted to and from the Web Server

Time Sheet	PM Role			TE Role		
	Sent	Receive	Total	Sent	Receive	Total
Portal	143.260	431.619	574.879	88.622	251.939	340.561
Portal, compressed	138.166	426.090	564.256	91.690	258.079	349.769
Java client	111.029	259.211	370.240	112.144	260.325	372.469

Expense Sheet	Portal, Compressed			Portal, Compressed		
	Sent	Receive	Total	Sent	Receive	Total
Portal	237.094	530.062	767.156	187.142	358.945	546.087
Portal, compressed	238.265	531.280	769.545	187.382	358.999	546.381
Java client	155.152	273.034	428.186	158.715	268.187	426.902

Scenario Printouts

Time Sheet Scenario

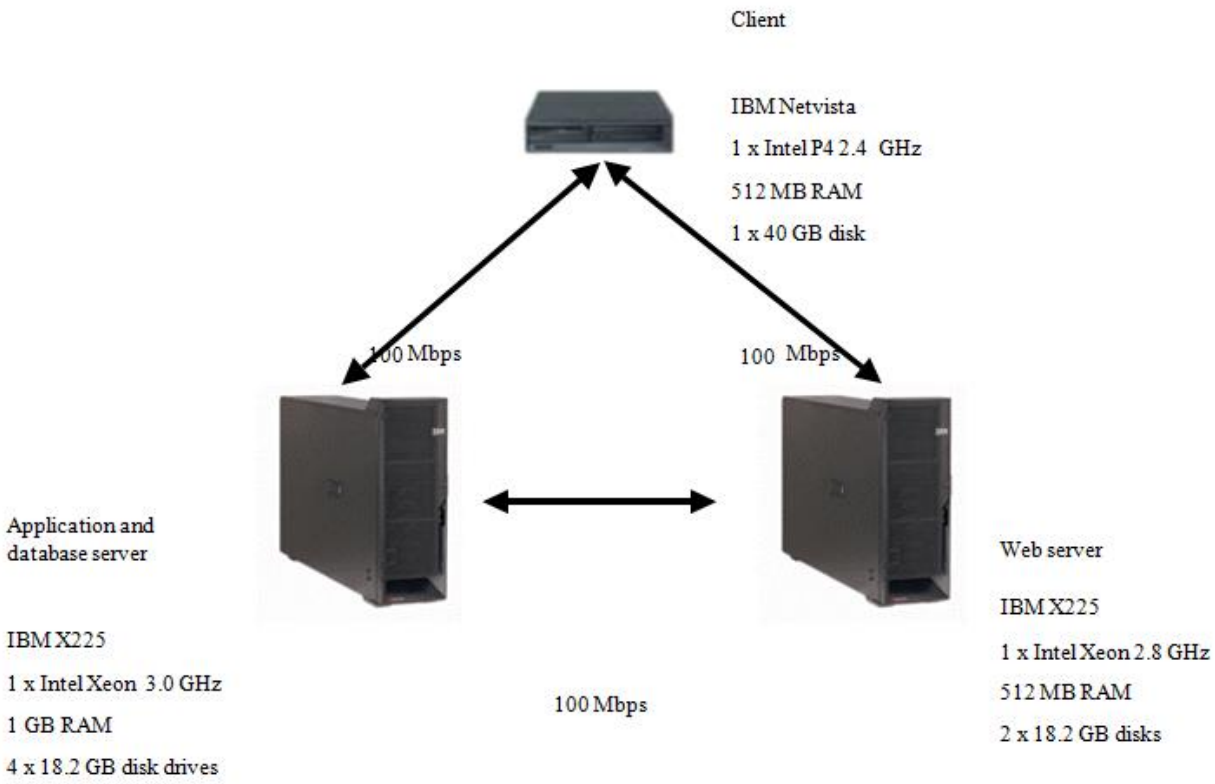
Time Sheet

Employee: Alan Lee
Employee No.: 1158
Week No.: 40
Year: 2005

Job No.	Job Name	Task	Description	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total
1410001	Waste Water - Bays	220	Counseling								
1310001	Due dilligence Britis	150	Analysis								
1110000	Administration	130	Internal time	8.0	8.0						16.0
1110000	Administration	140	Internal time			8.0					8.0
1110000	Administration	150	Internal time				8.0				8.0
1110000	Administration	160	Internal time								
1110001	Vacation	120	Vacation								
1110001	Vacation	110	Illness								
1310001	Due dilligence Britis	200	Counseling								
1310001	Due dilligence Britis	190	Counseling					10.0			10.0
Total				8.0	8.0	8.0	8.0	10.0	0.0	0.0	42.0

Hardware Setup

The following diagram shows the setup of the hardware used for the network load tests.



Software

The two IBM X225 servers are installed with Windows 2003 Standard Edition as the operating system.

Database and Application Server

- Windows 2003
- Service Pack 1a
- Oracle Version 9.2.0.6
- Maconomy Version 9.0 Service Pack 1

Web Server

- Windows 2003
- Service Pack 1a
- Apache 2.0.54

Integrations

Integration with other systems is most often performed by the Deltek Services team for specific customers. Integrations have during many years been implemented using a Web Services interface, by writing Mscript functions to handle the data exchange.

The WebService and Mscript will eventually be replaced by a RESTapi, which is available from Maconomy 2.2 and onwards. The termination of support for WebServices and Mscript is not scheduled yet, but new integrations should use the new RESTapi to be forward-compatible.

Installation and Upgrade Overview

Maconomy systems are often installed with one customer only in an installation. A customer may have multiple companies in the same Maconomy instance, all adding up to a consolidated group statement. This is supported by Maconomy, using the multi-company features.

Standard Maconomy System

The term “Packing Units” (PUs), is used by Deltek to describe the packages in which the entire application is distributed, before installation.

The architecture of a Maconomy installation roughly consists of a hardware platform with an operating system and a database installed. The Maconomy application consists of a platform-dependent Technology layer and a platform-independent Application layer. The Technology layer provides connectivity to the database and operating system facilities. The Application layer consists of the application business logic.

Both layers are installed using an installation program called MConfig. In addition to these packing units, there is a documentation package, which is also installed using MConfig.

The PUs reflect this architecture:

- TPU — Technology Packing Unit
- APU — Application Packing Unit
- SPU — Solution Packing Unit
- IPU — Installation Packing Unit
- DPU — Documentation Packing Unit

When new features or error corrections are released, they are released as installable PUs. They can be installed separately; you can, for example, upgrade the Technology layer (TPU) to exploit new server features without upgrading the application (APU). You should, however, observe any preconditions or special considerations as listed in the release documentation.

Release Documentation

For every new release of Maconomy (“service pack” or full version), Deltek issues a set of release documentation. This documentation describes new features and error corrections in the APU, TPU, or SPU respectively. You should always study the release documentation carefully before upgrading.

Application Server

The Application Server consists of two parts:

1. Original Maconomy Server (Legacy Server), which has been maintained in versions prior to 2.X.
2. New server components (Server 2.X), which contain newer server components required by the Workspace Client, for example, but also components that have been updated or renewed.

Work is ongoing to renew server parts, and updated parts will to a large degree be moved to the Server 2.X. The renewals of server parts is scheduled to obtain the goal of increased scalability and increased reliability.

Legacy Server

The Legacy Server is installed using Mconfig.

The server can be tuned and optimized if required.

Server 2.X

The Server 2.X is also installed using Mconfig.

Setting Up Coupling Services

In MConfig

On the Application screen, click the **OSGi products** button to start the setup.

The setup works much like the setup of web products: First select an existing server or create a one, which can either be the local machine or a remote server.

Selecting a server leads to the OSGi Products screen. As with web servers, the first time that the server is accessed, a screen with server parameters is automatically displayed, allowing you to select the root directory of the installation. This would typically be:

```
<MaconomyHome>/OSGi.W_15_0.<ApplicationName>
```

The main OSGi Products screen enables you to set a number of parameters. Currently, there are a number of logical groups:

- TPU and port numbers. The TPU is used both for the coupling service and update sites.
- Server process and memory setup.
- Business Objects connection setup.
- Single sign-on (SSO) information.

The meaning of these settings is described in the coupling service documentation.

Additionally, update sites for automatic update of workspace clients can be set up here.

Update sites for the Workspace Client reside on a web server. Web servers can be set up as usual from the Web Products screen. On the OSGi products screen, a selection list shows available web servers for use with Workspace Client update sites. Selecting web servers sets up update sites that allow Workspace Clients to update themselves when new versions become available in later TPU distributions.

Additional packages to be installed in the client, such as documentation or branding plugins, can be placed in the PUs folder. Available packages to be put on the update site can be selected on the OSGi Products screen. Packages must be valid P2 (the update site technology) repositories placed in a .tar archive named like MyPackage***-repo.tar***. See “Setting Update Sites” for Coupling Service with MConfig.

Note:

- Selecting a new TPU backs up current configuration files, but overwrites all changes that were entered directly in the files. Likewise changing any other setting for the coupling service causes MConfig to back up and overwrite a number of files, specifically server.ini and files in <serverroot>/Couplingservice/configuration/ and <serverroot>/Couplingservice/servicewrapper/conf/
  Any such changes must be restored manually.
- Web server and OSGi server names must be unique. Do not create an OSGi server with the same name as a web server.
- Coupling services that are created with TPUs prior to 14.0 sp 5 are incompatible with TPU 14.0 sp 5 and later. Such coupling services should be uninstalled before installing TPU 14.0 sp 5 (deselect Use Coupling Service on the OSGi Products screen in MConfig).
  On Windows, make sure that the service is removed from the Services control panel. If not, delete it using a command line like the following:

```
sc delete <service name>
```

The service name can be found by opening properties for the service in the control panel.

Proxy with Server 2.X

Client Server Interactions

In this document the following diagrams show the interactions between a client and a server:

```
                    Maconomy
Client            2.X Server
|                           |
|  HTTP                     |
|  GET /resource            |
|+------------------------->|
|<-------------------------+|
```

This diagram shows a setup where the client connects directly to the Maconomy 2.X server (formerly known as the Coupling Service) program using the HTTP protocol and receives a response.

Handshake

The Workspace Client and Maconomy 2.X have a protocol where the client, as the very first thing when connecting to a system, acquires some of its technical configuration from the server before proceeding with the interaction. This is a web service called the "server handshake."

```
                    Maconomy
Client            2.X Server
|                           |
|  HTTP                     |
|  GET /handshake/1         |
|+------------------------->|
|<-------------------------+|
|                           |
|  TCP 14444 unencrypted    |
|  DIRMI-RPC                |
|+------------------------->|
|<-------------------------+|
```

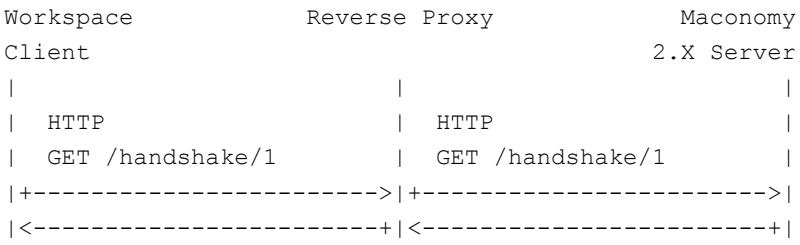
The purpose of this mechanism is to allow the configuration of a number of connection parameters on the server and have these options automatically distributed to the client transparently.

The following are some important parameters that are sent in the handshake:

- The transport protocol that is used for the Dirmi RPC interface: raw TCP sockets, WebSocket
- The mode of encryption: SSL encryption by intermediary, custom-made SSL encryption locally in the 2.X server, no encryption (not recommended)
- The mode of compression: custom-made zip block encryption, no compression (not recommended)

Intermediary Proxy

In some situations, the best practice is to introduce an intermediary proxy between the client and the origin server.



The terms “proxy server” and “origin server” are used to distinguish between the two servers.

In this setup the proxy server is technically a reverse proxy because the client connects to the proxy server that, seen from the client, transparently acts as the Maconomy 2.X server. In other words, the client knows only about the proxy server and considers it to be the Maconomy 2.X server.

The proxy server and origin server can be deployed on different physical machines, but they can also be run on the same physical machine.

Why Use an Intermediary Proxy Server?

The following are some common reasons for deploying a proxy server:

- SSL termination — Offloading the job of encrypting and decrypting traffic between the client and the server away from the origin server.
- Load balancing — Distributing requests from clients to a number of origin servers.
- Caching — Speeding up access to resources by caching resources at the proxy server, freeing up system resources at the origin server.
- Compression — Offloading the job of compressing the traffic away from the origin server.
- Translation — Offloading the job of translating resources to another language away from the origin server.
- Monitoring and filtering traffic.

SSL Termination

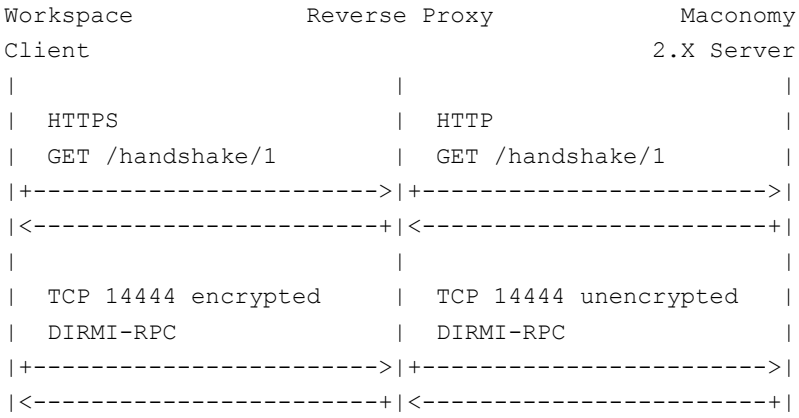
SSL termination is when the proxy server encrypts and decrypts the traffic between the client and the origin server. The proxy server communicates unencrypted with the origin server.

This is done to offload the task of encryption from the origin server.

Offloading encryption from Maconomy software to widely used standard software and hardware has other benefits:

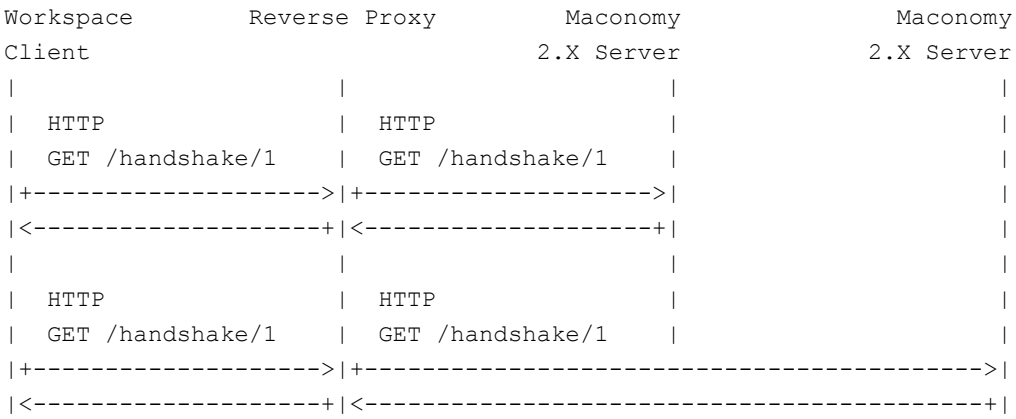
- Increased security because the proxy servers are widely used and actively maintained by companies and/or open-source communities.
- Setup and configuration are well known in customer organizations and fit well with existing infrastructure and network architectures.
- Managing server keys and certificates can be easier when using an SSL termination proxy.

Because the traffic to and from the origin server is unencrypted, it is very important that the origin server is secured, and that the channel between the reverse proxy and the origin server is secured. The origin server must not be generally accessible on the network because that could allow client programs to send data in an insecure way.



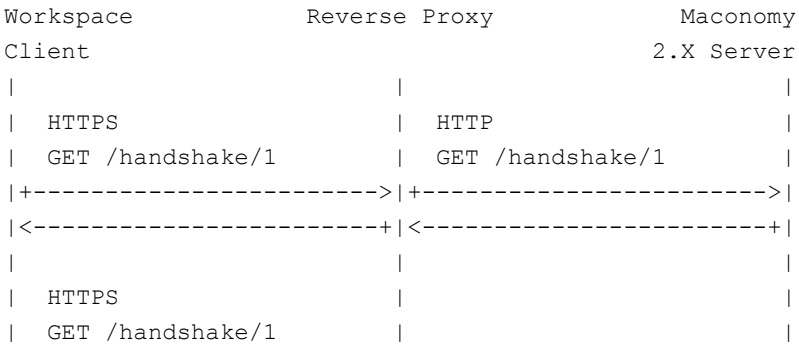
Load Balancing

Load balancing is when the proxy server distributes requests among a number of origin servers. This is done to scale the system to be able to support a higher throughput and thereby a higher number of concurrent users.



Caching

Caching is when the proxy server stores a copy of a resource and serves the copy without contacting the origin server. This is done to offload the origin server from generating or serving commonly used resources that change infrequently.



```
|+------------------------>|                          |
|<------------------------+|                          |
```

Setting up an Intermediary Proxy

There are several types of traffic to proxy in the Maconomy 2.X server:

- HTTP traffic
- Dirmi RPC traffic over either TCP or HTTP/WebSockets

Maconomy **always** exposes HTTP traffic to publish the server handshake and other web services.

The Dirmi RPC traffic is an interface that is specific to the Workspace Client, and it constitutes most of the communication between the Workspace Client and the Maconomy 2.X server. This interface can be set up with two different transport protocols:

- TCP sockets
- HTTP/WebSockets

The way that you set up a proxy differs, depending on the configuration:

- HTTP traffic and Dirmi RPC over HTTP/WebSockets require a standard HTTP proxy.
- Dirmi RPC over TCP requires a TCP proxy.

HTTP proxies generally offer a large feature set (caching, compression, SSL termination, and so on). Well known HTTP proxy servers include:

- The Apache HTTP server
- nginx
- Hardware proxies, such as F5
- Cloud proxy services when Maconomy is deployed at a cloud hosting provider such as Amazon

TCP proxies are less widely used, because many of the reasons for deploying a proxy server depend on the specific TCP application. Things like caching, compression, and translation cannot be done uniformly across all TCP applications. It is, however, possible to offer SSL termination using a TCP proxy server. Known TCP proxy servers include:

- `tcptunnel` (homepage) (GitHub)
- `stud` (GitHub)
- `nginx_tcp_proxy_module` (GitHub)
- Hardware proxies, such as F5

Dirmi RCP over TCP Sockets

In this mode a separate port on the Maconomy 2.X server accepts and handles the Dirmi RPC traffic.

Advantages:

- This is the original configuration for Maconomy 2.X, so there is more experience with the performance and stability of this configuration.

Disadvantages:

- This mode requires a separate port to be exposed to handle the Dirmi RPC traffic. This must be configured in intermediaries such as proxies and firewalls.
- In this mode, it may be necessary to set up two different proxy server software products: one for handling HTTP, and one for handling Dirmi RPC. A standard HTTP proxy server may not be applicable to both types of traffic.

Examples:

- In the following example (see “TCP Reverse Proxy – Example”), The TCP proxy servers stud and tcptunnel are set up for testing TCP tunnels.

Dirmi RCP over WebSockets

In this mode a web service endpoint on the Maconomy 2.X server accepts and handles the Dirmi RPC traffic. This is achieved by "upgrading" the HTTP connection to a full-duplex WebSocket.

Advantages:

- A standard HTTP proxy server can be used, but may require additional configuration in older software.
- Only one port needs to be exposed on the server, intermediaries, and firewalls.

Disadvantages:

- This is a newer feature in which many organizations do not have as much experience with performance and stability as for plain TCP sockets.

Setting Up SSL Termination

The Maconomy 2.X server must be configured to use SSL termination. When the 2.X server is configured in this mode it instructs the Workspace Client in the server handshake to require encryption, even though the origin server does not perform any encryption. Encryption must then be performed by an intermediary.

Examples:

- Please see the following example, Setting Up Proxy SSL on Server 2.X.

Generating Private Key and Certificate

If an organization does not have an official certificate, openssl tool can be used to generate one that is self-signed. In case of the tools used PEM formats were required. To generate the mykey.pem private key and mycert.pem certificate, run the following command:

```
openssl req -x509 -days 365 -newkey rsa:2048 -keyout mykey.pem -nodes -out
mycert.pem
```

Consult the manual for details about the additional parameters and their meaning.

Conversions between the Certificate File Formats

It might be the case that a proxy server requires the certificate and key in a different format. The following are conversions to pkcs12 and jks if they are required.

PEM to PKCS12

Input: a certificate file (mycert.pem) and a private key file (mykey.pem)

Command:

```
openssl pkcs12 -export -in mycert.pem -inkey mykey.pem -out mycert.pkcs12
```

Result: PKCS12 formatted mycert.pkcs12 file

PKCS12 to JKS

Input: a certificate file (mycert.pkcs12). Requires a keytool that is supplied with the JDK.

Command:

```
keytool -importkeystore -srckeystore mycert.pkcs12 -srcstoretype PKCS12 -
destkeystore mycert.jks -deststoretype JKS
```

Result: JKS formatted mycert.jks file

TCP Reverse Proxy – Example

This section describes how to set up a TCP reverse proxy that forwards to a Maconomy 2.X server. This is useful for running a reverse proxy when the coupling service is configured to use TCP as its transport protocol.

These examples use a reverse proxy machine and a Maconomy server machine. In this setup these are separate machines, but they can also be the same machine.

In this example the reverse proxy server machine has the IP address 172.16.1.115, and the Maconomy server machine has the IP address 172.16.1.150.

A client connects to the reverse proxy machine and connections are then forwarded to the Maconomy server machine.

In these examples the handshake is SSL-terminated on the proxy server machine using an HTTP proxy (nginx), but this is not required.

Plain TCP Reverse Proxy

A plain TCP forwarding proxy does nothing but forward TCP connections to another process or another machine. No SSL termination, caching, or similar actions are performed.

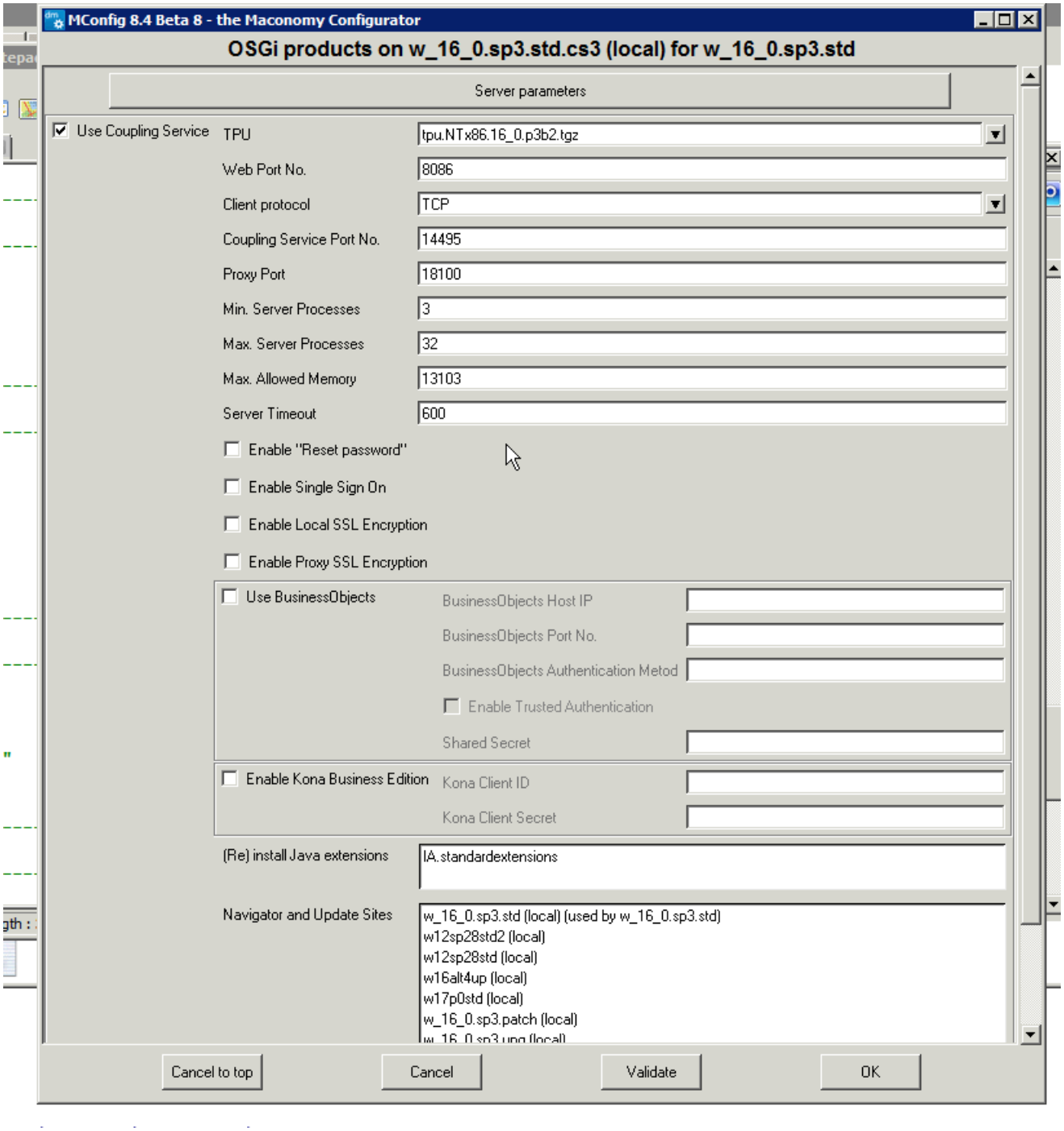
```
Workspace             Reverse Proxy              Maconomy
Client                                         2.X Server
|                           |                           |
|  HTTPS                    |  HTTP                     |
|  GET /handshake/1         |  GET /handshake/1         |
|+------------------------->|+------------------------->|
|<-------------------------+|<-------------------------+|
|                           |                           |
|  TCP                      |  TCP                      |
|  Unmodified 18100         |  Unmodified 14495         |
|+------------------------->|+------------------------->|
|<-------------------------+|<-------------------------+|
|                           |                           |
```

This example uses tcptunnel (homepage) (GitHub).

We run the program with the following parameters:

```
./tcptunnel --local-port=18100 --remote-port=14495 --remote-host=172.16.1.150 -
-stay-alive --bind-address=172.16.1.115 --fork
```

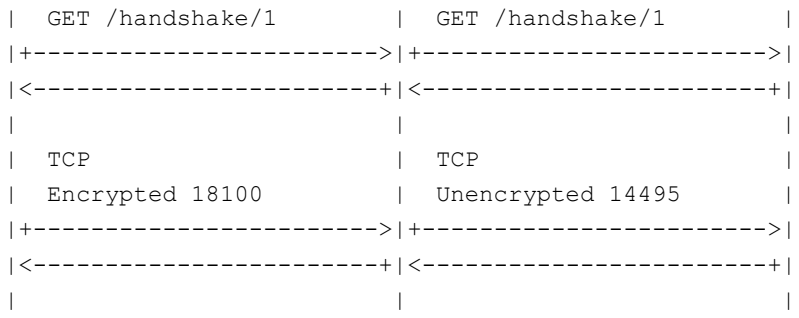
In MConfig, set up the coupling service to use TCP with either no encryption or local encryption:



SSL-Terminating TCP Reverse Proxy

An SSL-terminating TCP reverse proxy provides encryption between the client and the proxy machine and forwards unencrypted data to the Maconomy server machine.

```
Workspace            Reverse Proxy            Maconomy
Client                                      2.X Server
|                        |                        |
|  HTTPS                 |  HTTP                  |
```



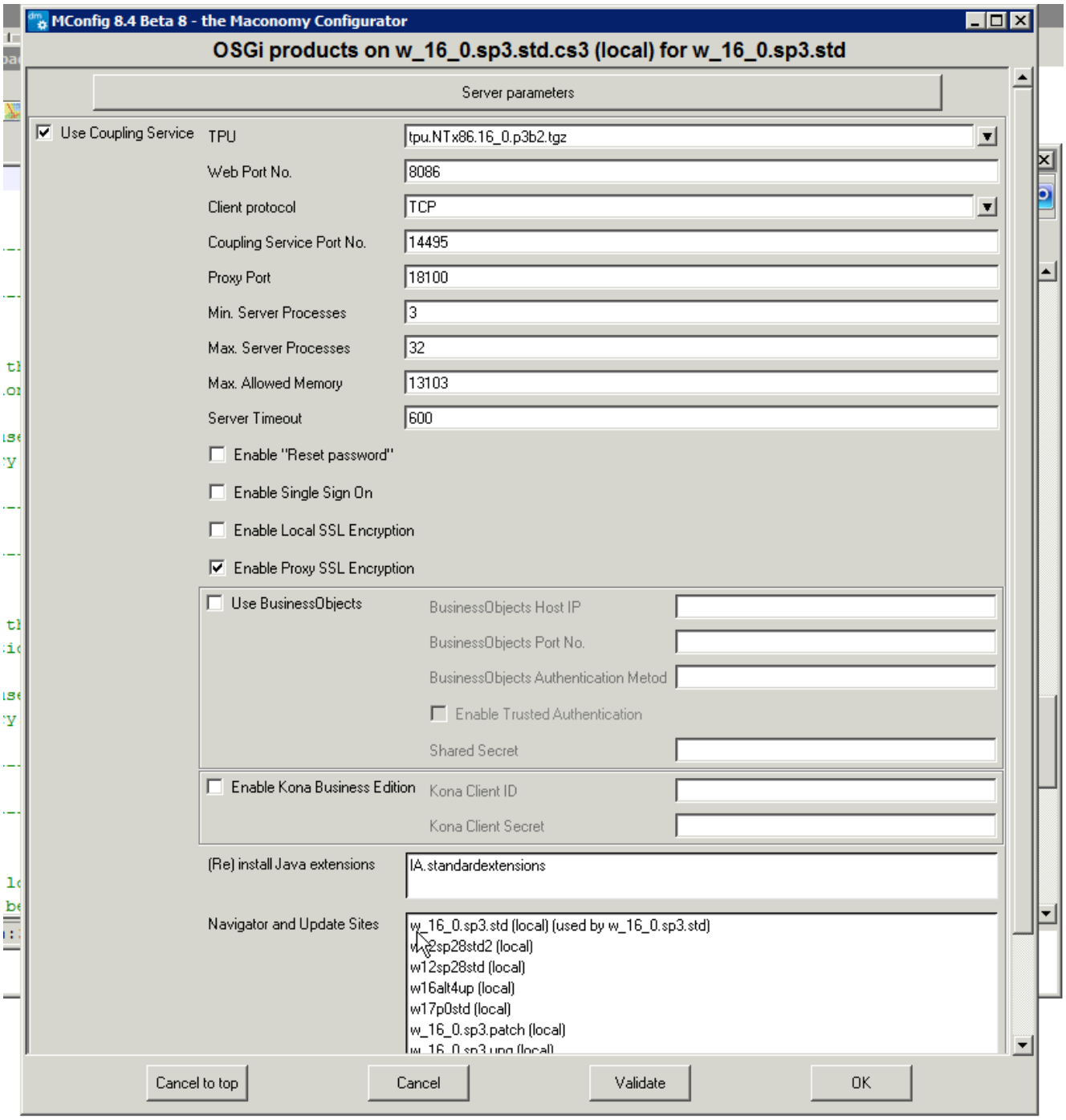
This example uses stud (GitHub)

Run the program with the following parameters:

```
stud --ssl -b 172.16.1.150,14495 -f 172.16.1.115,18100 big.pem
```

where `big.pem` is a PEM file that contains the server certificate and private key. This must be the same certificate and private key used in the HTTPS handshake.

In this example, in MConfig the coupling service is set up to use TCP with proxy encryption:



Setting Up Proxy SSL on Server 2.X

To configure the proxy SSL, the new server.ini Boolean option was created and named coupling.dirmi.proxy.encryption. By assumption it cannot be set together with coupling.dirmi.encryption, because doing so would cause failure on the Coupling Service start-up. There is an optional server.ini parameter called coupling.dirmi.proxy.port to be used in situations where the proxy server is exposed on a port other than the Dirmi port of the Coupling Service

(specified in server.ini by coupling.dirmi.port). In this case, specifying coupling.dirmi.proxy.port hides the coupling.dirmi.port specification when creating the handshake.

Example of the correct server.ini specification is the following:

```
coupling.dirmi.proxy.port = 14444
coupling.dirmi.encryption = false
coupling.dirmi.proxy.encryption = true
```

This configuration specifies that the proxy SSL (through coupling.dirmi.proxy.encryption) should be used, and the regular SSL is turned off (through coupling.dirmi.encryption).

Web Daemons

The Maconomy Web Daemon connects web server components to the Maconomy server. When a web component, such as M-Script or the User clients, requests data from the Maconomy server, the Web Daemon handles the request and forwards it to an available server process.

When the Web Daemon process is started, it creates a pool of Maconomy server processes. Maconomy is then ready to deliver requests from web clients, such as the Portal or Java Client.

BPM

BPM is the reporting solution for Maconomy. BPM is installed and configured using Mconfig and contains a set of standard reports. The reports can be viewed/printed from different clients, both the older Portal and the newer Workspace Client.

Each customer may have special reporting requirements that may require special adjustments or customized reports. These are developed either by the customer or by consultants.

Clients

A Maconomy system can be accessed from different User Interfaces or Clients. Customers will be advised which user interface best suits the needs of their Users based on their different user roles. For example, financial and accounting users will prefer a client that supports heavy data processing and complex workflows, such as the Workspace Client. Front office Users who mainly work remotely at customer locations, such as consultants, may prefer a web-based client, such as iAccess or the legacy Portal, or a client optimized for mobile, touch-screen devices, such as Touch.

See iAccess and Touch documentation for more information on those products.

Workspace Client (WSC)

The Workspace Client is available for both Windows and for Macintosh users. Each Maconomy release contains a TPU package which in turn contains a WSC package containing a ZIP file, a Windows-installable package (.MSI), and a Macintosh-installable package (.DMG).

Deltek Best Practice

Windows Installation

- From DSM, download update version ZIP file, such as:

  DeltekMaconomyWSClient24LA2.zip

- From this ZIP file carefully locate the contained file, such as:

  Maconomy-20_0.p001e-dirmi-win32.win32.x86.zip

- Extract this file from the ZIP file and install the Workspace Client on the user's machine.

Update Sites

Using the ZIP packages enables you to use Update Sites, ensuring that new versions of the Client, translations and branding (if required) are automatically downloaded and installed on User's machines whenever they are available. All updates to the WSC released via new major or minor versions, maintenance releases or cumulative updates can then be automatically installed via Update Sites rather than manually.

If you upgrade to a new version of Maconomy without enabling full Update Site functionality, you have to uninstall the previous version prior to installing a new WSC.

MSI Installer is NOT Recommended Method

Workspace Clients installed using the MSI installer cannot use Update Sites and are **not** automatically updated for all upgrades. Since Update Sites require write access to the directory where the Client is installed, Workspace Client upgrades where the embedded Java is changed only update automatically if installed using the ZIP packages.

If the WSC is installed using this method, Deltek cannot guarantee that Update Sites will support a new version of the WSC, and this may require that a completely new installation is made for each user.

Deltek Best Practice

Macintosh Installation

- From DSM, download update version ZIP file, such as:

  DeltekMaconomyWSClient24LA2.zip

- From this ZIP file carefully locate the contained file, such as:

  Maconomy-20_0.p001e-dirmi-macosx.cocoa.x86_64.dmg

- Extract the DMG package.
- Install the Workspace Client on the user's desktop or in the user's Home folder in order to allow read/write access. This ensures that the WSC automatically updates in all scenarios.

Do NOT Install in Global Applications Folder

Installing in the Global Applications folder allows only read-only access to the WSC, which prevents automatic updates and is likely to require that updates to the WSC will require a new installation.

Prepopulate IP Address and Port Number

When installing the Workspace Client, you can prepopulate the IP address and Port number in the .ini file so that the end-user does not need to enter them at first login. Update Sites overwrite the .ini file when the Client is updated. If you would like to prepopulate IP addresses and Port numbers for end users and would like this to be maintained post-upgrade, you must implement a .cmd wrapper to launch the WSC, such as:

```
Maconomy.exe -a <hostname> -p <portnumber>
```

Windows Uninstall

To uninstall the Workspace Client on a Windows computer, complete the following steps:

1. Start the Windows Control Panel.
2. Select **Uninstall a Program**, or go to **Programs and Features » Uninstall or change a program**, depending on the version of Windows that you are using.
3. Select the Deltek Maconomy Workspace Client.

   The Windows uninstaller removes the Workspace Client.

Macintosh Installation and Uninstall

To uninstall the Workspace Client on a Macintosh computer, complete the following steps:

1. Locate the .DMG file.
2. Click the **.DMG file** and drag the package into the preferred location.

   The Macintosh uninstaller provides prompts for the steps that are required.
3. Follow the steps and respond appropriately.

   The Macintosh installer removes the Workspace Client.

Analyzer Overview

The Workspace Client allows opening the Java Analyzer window directly from the Workspace Client so that all of the Analyzer capabilities and reporting functionality is also available in Workspace Client.

In addition, you can define which specific report to open in Java Analyzer, and perform an automatic user login to Java Analyzer without the user having to manually enter credentials every time.

Java Analyzer window is the same as the standard Java Client, which contains all of the functionality that is related to the Analyzer reports.

The Maconomy Server for Oracle allows you to use query-specific Oracle "hints" when executing SQL for Analyzer reports, filters, value pickers and notifications.

Workspace Client Analyzer

Java Analyzer is built into the Workspace Client and can be embedded in any MDML layout. The Java Analyzer uses the MaconomyWeb login rule for automatic user login without the user having to enter credentials, after an initial specification. In addition, the Java Analyzer uses the URL for connecting to the Maconomy Server, which also must be specified.

Analyzer MDML Specification

The Java Analyzer can be opened via MDML Report tag, similar to the Business Objects reports.

The current MDML specification for the <Report> tag has not changed, except for a new attribute engine that defines which external system, like Business Objects or Analyzer, to use for reporting.

In the current implementation, you can only use Action <Report>-tag to run the Analyzer report, unlike Business Object reports that can also be embedded in the View.

You can use the <Report>-tag to open an Analyzer report in a new window. The <Report>-tag can open a Java Analyzer in a new window, or a specific Java Analyzer report if the view attribute specifies the name of the report to open.

In addition, the detailed specification of the <Report>-tag in the Report Action is described in the following.

- If the all-attribute on the <Actions>-tag is **True**, the <Report>-tag overrides individual properties of the Report action.
- If the all-attribute is **False**, the <Report>-tag is also used to include this action.

Attribute Name	Type	Usage
name	Key	The unique name of this action. Use this to distinguish actions with the same source. If no name is specified, the name defaults to the same name as the source. This attribute is general for all Actions.
ref	Key	Use the ref-attribute in the context of the <Order>-tag to refer to previously defined actions during re-ordering. The ref-attribute is the only legal attribute in this context. This attribute is general for all Actions.

icon	Id	The resource identifier for an icon to be associated with this action. This attribute is general for all Actions.
title	Display	The displayable title of this action. If nothing is specified the action receives its title from the server specification. This attribute is general for all Actions.
appearance	(Action Appearance Type)	The visual appearance of an action determines how it is rendered in the user interface. This attribute is general for all Actions. Determine if the title, icon, or both should be displayed. Valid values are: ▪ Standard ▪ All ▪ Title ▪ Icon
tooltip	Display	Indicates a tool tip title that is displayed when the mouse hovers above the action. If nothing is specified, the tool tip is the same as the title. This attribute is general for all Actions.
engine	(Engine Type)	Indicates the engine type of the report. This attribute is specific for the Report Action. There are two different types of reports supported: Business Objects report and Analyzer report. Valid values are: ▪ businessObjects ▪ analyzer This attribute is optional and the default value is businessObjects. If no value is specified the Business Objects report is opened, to support all existing layouts.
source	Expression (String)	Indicates the source of the report. This attribute is specific for the Report Action. The precise definition of the source depends on the third-party system that produces the report. This attribute is required, and the default value is empty string. This attribute is not used for Analyzer report at the moment.
view	Expression (String)	Indicates the view to be used from the source of the report. This attribute is specific for the Report Action. The precise definition of the view depends on the third-party system that produces the report. The default value is the empty string. For the Analyzer report this defines then name of the specific report.
output	(Output Type)	The output format of the report. This attribute is specific for the Report Action. You can use a report action to open a report inline or in an external

		window. The default format of the report is HTML. Valid values are: ▪ html ▪ pdf ▪ xls This attribute is not used for Analyzer report.
preTrigger	Key	The identifier of the trigger associated with this report action. This attribute is general for all Actions. When you specify a preTrigger, invoking the action calls the trigger to run before the action is executed. ▪ If the trigger runs successfully, the action is executed normally ▪ If the trigger fails, the action is never executed and the error message is displayed.

Analyzer Login Rule Specification

The login rule for the Analyzer should be specified in Coupling Service configuration similar to the Business Object login rule. Add the following to the `maconomy.security.config` file:

```
MaconomyWeb{
  org.eclipse.equinox.security.auth.module.ExtensionLoginModule required
    extensionId="com.maconomy.lib.coupling.MaconomyWebLoginModule";
};
```

This allows users to log in to Java Analyzer automatically from Workspace Client. For details about the login process see “Login Handling.”

Analyzer URL Specification

To use Java Analyzer the URL to the web server for Maconomy should be specified. The URL is database- and language-specific. MConfig handles this process automatically. The URL is stored in the `analyzer.json` file, which is distributed similar to `maconomy.security.config` file. The following shows the template `analyzer.json` file.

```
// Substitute the DATABASE_SHORTNAME, LANGUAGE and URL to match your web server
[
  {
    "database" : " DATABASE_SHORTNAME ",
    "urls" : [
      {
        "language" : " LANGUAGE ",
        "url" : "HOST:PORT/cgi-bin/Maconomy/Jaconomy.DATABASE_SHORTNAME.LANGUAGE.exe"
      }
    ]
  }
]
```

MConfig automatically replaces the `DATABASE_SHORTNAME`, `LANGUAGE`, and `URL` to match the current Web Server.

The Workspace Client reads the `analyzer.json` file from a web service, which will be started by the Coupling Service. The Workspace Client will query specific URL for the currently running database and language.

In addition to that, this analyzer web service is a simple http service, and therefore you can run the queries directly in the browser. This allows users to check whether the Analyzer is configured correctly on the Coupling Service.

For example, after the Coupling Service is started the user can run the following query in the browser:

```
http://<coupling_service_host>:<port>/analyzer/v1?database=<value>&language=<value>
```

The result of this request should be a valid URL to the web server where the Java Analyzer is installed, which is database- and language-specific.

Java Analyzer

Java Analyzer is a special mode of Java Client where only a limited set of functionality is available to the user. Java Analyzer starts up in a separate window and includes the top menu to open specific Analyzer Report, View/Edit menu, and quit the Analyzer.

Java Analyzer Internals

Java Analyzer is a special mode of Java Client, where the following features are removed based on the Analyzer mode:

- Removed access to Preference window tabs that are not relevant to the Analyzer
- Removed retrieval of global dialogs list and menus
- Removed Java Client splash screen
- Removed task bar icon
- Removed access to dialog windows and dialog tabs
- Removed "Menu" and other dialog-specific menus
- Removed access to notifications
- Removed access to "RGL Reports…" and "Programs…"
- Removed WebStart functionality
- Removed auto start functionality
- Removed dialog help contents from help system, except for Analyzer-specific help
- Removed creation of desktop icon
- Removed non-Analyzer Windows menu items
- Removed access to Menu in RGL / MSL programs

Instead, Java Analyzer starts up in a separate Java process and has direct process communication with Workspace Client, which is a parent process for Java Analyzer.

Shared Format Preferences

In addition the Java Analyzer shares Format preferences with Workspace Client for the dates, time, currency, integers and decimals. For example if the user changes any of the Formatting Preferences in Workspace Client, the changes are reflected automatically when the user opens a new Analyzer report.

Analyzer Start and Quit

Java Analyzer can be started from a Workspace Client. Java Analyzer starts up in a separate Java process suing JRE that is already embedded in the Workspace Client, and has a direct process communication with a Workspace Client. Every time the user opens a new Report the Workspace Client can determine if the Java Analyzer is already started and open Report in the same Java process. If the Java Analyzer is not started or was closed previously, the Workspace Client starts up a new Java Analyzer process again.
The user can also quit the Java Analyzer independently of the Workspace Client, using the Quit menu option. Closing the Report window in Java Analyzer does not quit the Java Analyzer, similar to the Java Client. Another Report can still be opened from the top Analyzer menu.

Java Analyzer shares the “dirty” state with the Workspace Client. For example, if the user quits the Workspace Client, but the Java Analyzer still has unsaved data, the user is notified and can save the data.

Login Handling

It is not necessary for the user to log in to the Java Analyzer. The login process is handled automatically for the user via the MaconomyWeb login rule, which must be specified on the coupling service, similar to the Business Objects login rule.

The MaconomyWeb login rule can produce a token, which is handed to the Workspace Client via principal. The token plus the user name can be then used by the Java Analyzer to connect to the server, validate the token, and get the access automatically. If the token cannot be validated, the user is prompted with the user/password window.

Java Analyzer URL

In addition to the login information and the name of the specific report to open in Java Analyzer, it is necessary to specify the URL to the web server of the Maconomy system, which is database- and language-specific.

Touch

Touch is a separate Deltek offering that can be used with Maconomy. Touch provides a client for mobile devices. Touch includes a limited set of capabilities, compared to the standard clients for Maconomy.

Maconomy iAccess

Maconomy iAccess is a separate Deltek offering that provides a client for Maconomy on touch and tablet devices. iAccess is an HTML interface to the Maconomy system and can be used with other clients, such as the Workspace Client. iAccess was introduced with Maconomy 2.2 and supports the most commonly used end-user features; functionality continues to increase in subsequent releases of iAccess.

Portal

Installation of the Portal is managed by Mconfig.

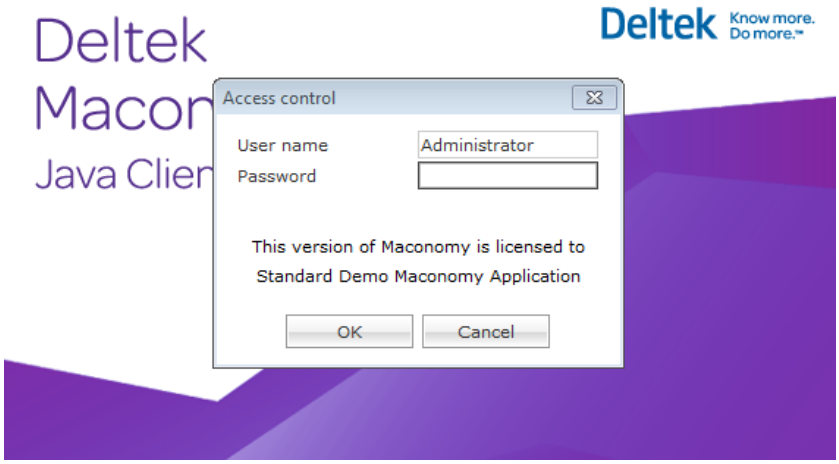
Installation of the Portal software is a server-side-only component, and Mconfig is capable of installing all relevant parts: web server, web daemons, and so on.

When installing the Portal the setup needs to be designed for the right scalability, for example, how many web servers are required, what about load balancing, and so forth. These items are components that are beyond the Maconomy delivery and can be configured using standard components from different vendors.

To configure the web daemons to perform optimally, see “Web Daemon” for more information.

Java Client

Installing the Java Client, named Jaconomy, requires that you locate the JaconomyMP/webstart.jnpl file. Click on the .jnpl file and installation is managed automatically. After installation you are presented with the login screen, shown in the following figure.



Update Sites

Update Sites are used to update/install additional features for Maconomy 2.1.5 (and newer) Workspace Client.

Automatic updates of the Workspace Client are possible in most installations, but some versions of the Workspace Client may contain components that cannot be updated and that may require re-installation of the Workspace Client. Some customers may also have restrictions on their computers (such as terminal servers) that may prevent automatic updates to succeed, and that thereby require re-installation. Windows access rights management may also prevent automatic updates to be successful if the Workspace Client was originally installed using the MSI installer. Deltek thereby recommends that installations of the Workspace Client on Windows platforms be performed using the ZIP packages.

The update/install is performed by the Workspace Client using the p2 provisioning platform, which requires the installable/updated units to be packaged as Update Sites and made available on a web server. If new Workspace Client features do not prevent automatic updates, Update Sites for Maconomy 2.1.5 (and newer) Workspace Clients are distributed with the Coupling Service and installed with the Coupling Service using MConfig.

The process of the install/update mechanism can be described as the following:

- The update repositories are created using Build Updatesite tool and delivered as independent packages `-repo.tar` files.
- The Update Sites are installed with the Coupling Service and deployed on a web server using MConfig.

- When the Update Sites are installed on the web server, the Workspace Client will be able to connect to the Coupling Service and acquire the location of Update Sites on the web server, resolve necessary dependencies, and download updates/installs from specified location.

This document describes how to set up Update Sites for the Coupling Service using MConfig.

Before You Begin

Before you begin, review these notes about Update Sites:

- Update sites can be used to update:
    - New versions of the Workspace Client, which comes based on a Maconomy update as a CU, MU or new version,
    - Dictionaries, for which customers implement changes
    - Branding, for which customers implement changes
- Updates of the WSC are typically of equal size as new versions of all plugins that are included in the update. However, updates to new versions may also include changes reserved only to new versions, which may make them slightly larger.
- Update Sites impact customer networks, including information that updates are at present passive until Users actively attempt to open the WSC following a new change. Consider this consequence if this is likely to happen at a single point in time, such as Monday morning after an upgrade. Also consider other 3rd party software distribution providers or local proxies that could be put in place to shadow the main update site, and therefore reduce the network impact. This would allow you to implement local scripts to push updates more efficiently to Users on the network, even if they do not actively open the WSC.
- For branding and dictionaries, no update is made unless changes to the timestamp are made. Therefore, if no branding changes are made, this does not add to the size of the update.

Building Update Sites

A Documentation package (Maconomy Help) is delivered as an Update Site. The package is created using Build Updatesite tool and packed in `<package_name><timestamp>-repo.tar` file. The following are the most important elements in the folder structure of the Update Site package:

- `/artifacts.jar` — The generated artifact repository index file for the Update Site.
- `/content.jar` — The generated metadata repository index file for the Update Site.
- `/plugins` — This folder contains the plugins that compose the Update Site and is downloaded during the update.
- `/features` — This folder contains the features that compose the Update Site and is downloaded during the update.

The Build Updatesite tool can be run with Ant if the input plugins and features to be exported into Update Site. The output of the tool is a generated Updated Site packaged in the `<package_name><timestamp>-repo.tar` file.

Installing Update Sites with MConfig

After the Update Site package is created, it is ready to be installed and deployed with MConfig. MConfig is intended to unpack the Update Site `-repo.tar` package and deploy it on the web

server. In addition, MConfig writes the location of the deployed Update Site into the `updatesite.ini` file on the Coupling Service.

The following are detailed instructions on how to install Update Sites with MConfig:

- Update Sites for the Workspace Client should be deployed on a web server. Web servers can be set up as usual from the Web Products screen.
- On the OSGi products screen, a selection list shows available web servers for use with Workspace Client update sites. To deploy Update Sites on a specific web server select if from the list
- Additional packages to be installed should be placed in the PUs folder. Available packages to be deployed can be selected on the OSGi Products screen. Packages should be valid P2 repositories placed in a .tar archive named like MyPackage***-repo.tar***.

For more details refer to “How to Install the Coupling Service with MConfig”.

Additional Step When Using IIS Web Server

Some of the Update Site files that must be downloaded by the client have no naming extension. If Update Site is set up on a web server that runs on IIS, these files are by default inaccessible.

To enable access, configure the IIS site to allow Extension “.*” with MIME as the type of “applications/octet-stream.” Set this up in IIS manager, on the “MIME types” configuration panel specifically for the Maconomy/Update Site path on the web site under consideration.

Updating the Maconomy Workspace Client

When Update Sites are successfully installed with the Coupling Service and deployed on the web server using MConfig, they are ready for Maconomy Workspace Client. The update process can be described as follows:

- The Update Sites are deployed on the web server and are ready to be downloaded.
- The Coupling Service knows about the location of the Update Sites from which they can be downloaded.
- The Workspace Client starts up and during the initial login process connects to the Coupling Service.
- The Coupling Service notifies the Workspace Clients about available updates/installs and provides the location for Update Sites.
- The Workspace Client checks and resolves possible update/installs and downloads them from the specified location.
- If new updates/installs were downloaded. the Workspace Client restarts so that the new changes take effect.
- If no updates were found, the Workspace Client starts up normally.

Solutions

As part of a Maconomy release, two solutions—PSO and CPA—are released as sub-releases. These two solutions can be installed on top of the core Maconomy by using Mconfig.

The two solutions are packaged in solutions files that are targeted for a specific Maconomy release.

Extensions

A Maconomy release contains some product extensions. Some are built into the core Maconomy systems and are seen as standard functionality, such as “export to Excel”, whereas other extensions are installed as separate functionality, such as “Credit Control”. The standard product extension should by default always be installed using Mconfig. In special customer solutions it may be decided not to install the standard extensions, and default selections in Mconfig may need to be changed when installing packages.

Configuration and Maintenance

A Maconomy system can be customized by just changing the configuration, for example, how security is configured, which protocol is used between Workspace client and Server, and so on.

For more information see the following descriptions.

Database Server

Maconomy can be installed on an Oracle or SQL database. The selection of database is often decided by the customer and is based on his or her preferences. The recommendation is to use Oracle if the customer does not have any strong preference. The list of supported database versions is made available with each Maconomy release.

Killing Processes in Oracle

Occasionally you will want to stop Maconomy from doing what it is doing. This typically happens if a user starts running a very large Analyzer report, for example, without setting any parameters at all. When the user realizes how large the report is, or if he or she discovers that it was the wrong report, the user will want to try to stop the report. The only way to do this, in the average user's eyes, is by terminating the Maconomy client.

However, since the processing of the report does not take place at the client, terminating the client does not end the process of generating the report. To stop a runaway process, you must terminate (kill) the process in the Oracle database.

This section describes the way in which this is done. Note that only persons who have a thorough understanding of the Oracle database should perform the steps outlined in this document.

The process of identifying and killing an Oracle process is very different depending on the platform. For this reason, the following description is divided into two sections: "Kill Process under Windows NT" and "Kill Process under Unix."

Kill Process under Windows NT

This section describes how a runaway process is terminated if your Maconomy server is running under Windows NT.

The OraEater Tool

It is important to identify the correct Oracle process before killing it. With TPU 34 (27.00), a tool called oraeater was released. This tool determines which Oracle threads are using system resources intensively (such as Analyzer reports that do not have limiting selection criteria). The tool usually resides in the /bin directory of your Maconomy server.

Usage

```
oraeater     <Oracle username> <password> [-a]
```

A list is displayed, showing which Oracle threads have consumed the most CPU power for the last 20 seconds. The -a option shows the most CPU-consuming Oracle threads in total. The current thread and Oracle system threads are not shown.

Example Output

If an Oracle-eating process was found, the output from the tool might look like the following (without the -a option).

```
-THREAD  PROCESS  USERNAM  PROGRAM      LOG_READS     Last 20 secs
-523     360:434  Claudia  sqlplus.exe  27237,26      17119,91
-Active SQL
-select count(*) from code a, code b, code c, code d, code e
```

The preceding output indicates that thread number 523 is consuming a large amount of Oracle resources. The Active SQL part of the output shows all currently active queries, not just those that are related to the shown threads.

Resolution

To kill the runaway process, use the Oracle tool orakill in the following way:

```
orakill <ORACLE_SID> <threadno>
```

Using the previous example, you can enter the following to terminate the problem process:

```
orakill ORCL 523
```

This terminates the process.

Kill Process under Unix

This section describes how a runaway process is terminated if your Maconomy server is running under Unix.

When a client logs in to Maconomy, and Maconomy resides on a Unix server, two processes are started:

```
macoracle.r <application_id> oracle<SIDname>
```

The <application_id> is the shortname of your Maconomy application, for example, W_7_0. An example of <SIDname> could be Orcl. The Parent Process ID (PPID) of <SIDname> is identical to the process ID (PID) of macoracle.r.

To identify the correct process, it is important to know whether the client that initiated the process has been terminated or not, that is, if the Maconomy client has been forced closed while, for example, running an Analyzer report that does not have limiting search criteria. If the client has not been terminated, you can use the same method as you use when the client has been terminated, but not vice versa.

Two-Tier or Three-Tier

In a two-tier Maconomy solution, the Maconomy server and the Oracle server reside on the same physical server. In a three-tier solution, the Maconomy server and the Oracle server reside on two separate physical servers. Throughout the following description, the terms “Maconomy server” and “database server” are used. The principles are the same, regardless of whether you are running a two-tier or a three-tier system.

Application Server

This document describes the options available for the Maconomy server executable, and describes the initialization files that influence how the Maconomy server and clients operate.

The main Maconomy executable, maconomyserver.exe (on the Windows platform) or macoracle.r (on Unix), supports a number of modes and features. These modes and features are controlled by server options.

The options can be used to specify a mode, such as which TCP port the Maconomy server should listen to, or a feature, such as forcing the Maconomy server to validate custom print layouts.

These options are described one by one in the first section of this manual. In addition, Maconomy uses initialization files to read options every time that the server or another application is started. These initialization files are located in the IniFiles subdirectory of the Maconomy home directory. The second section of this manual describes the initialization files: what they apply to, what you can enter in the files, and when they are read by the Maconomy server application.

Options

This section describes the options that are available to the Maconomy server executable (maconomyserver on the Windows platform; macoracle on Unix). The following table provides an alphabetical overview of all server options.

Command-Line Options

	-DM<f>	M-Script-related	-t
-A<dir>	-DS<f>	-P	-U<x>
Analyzer-related	-E<f>	-pC	-v
-b	-e<v>	--port <portNo>	-vv
-c<n>	-f	-Q	Workflow-related
-C<x>	-FG	-r<n>	-W
--CreateDocumentTables	-FV	-R<r>	-w<x>
-CS	-h	-s<c>	-x<p>
-d	-i	-S<n>	-Z<r>
-D<x>	-I<f><!>	-socket <s>	
--DDLBuffers<n>	--IP<str>	-T	

Command line options are entered on the command line on Windows or as a Unix shell command. The command has the following form:

`Maconomyserver -[-]option[]<argument>`

Note that some options have one leading - (hyphen), and some have two (--). Also, some options require a space between the option and the argument, whereas most require the argument directly after the option. These differences are all reflected in the descriptions below.

Server Operation

Query and/or manipulate the user session table as needed.

Options can be chained and are processed in the order they appear. The options 'time' and 'extended' must appear before the options to which they apply.

The following string abbreviations are permitted:

exp(ired), abs(olute), rel(ative), int(ernal), ext(ended)

Examples:

--users:time=relative:list List all user entries, relative time

--users:ext:list List all user entries, extended format

--users:list=exp:remove:list List and remove expired user entries, then list all (remaining) user entries

Option	Description
-f	Start the server in foreground mode. The server listens by itself instead of using the inetd (UNIX) or MaconomyDaemon (Windows).
-h	Displays a help page with a brief explanation of each option in the Maconomy Server.

-I	Prefix (basename) of .I file used when running (Windows only). Example:
	`MaconomyServer -imaconomyserver.W_9_0`
	Note that it is not necessary to include maconomyserver in the argument for this option. Hence, the preceding command can also be entered this way:
	`MaconomyServer -iW_9_0`
	In this case, the Maconomy server uses the initialization file called maconomyserver.w_9_0.I for reading startup options.

-socket <s>	Specify the socket to listen to (Windows only).
-v	Show the current version string. Result (example): `MACONOMY SERVER for ORACLE vers. 33.02.0.330252 compiled Mar 3 2004 11:16:48`
-vv	Show the current version string in extended format. This command lists the version numbers of the Maconomy server and other components such as MScript and MQL, and lists various other information regarding the current release of Maconomy Server.
-S<n>	Set the shortname to <n> (avoid prompt).
--port <portNo>	Make the server listen on TCP port <port no.>. This option is ignored if the option -f is not used.
--DDLBuffers <n>	The server caches `<n>` DDL files (Dialog Definition files). The default value is to cache 10 files.
-x<p>	Run the script/container specified by `<p>`. `<p>` can be a standalone program, a container, or a precompiled report. The server automatically detects whether `<p>` is invoked for interactive execution or for batch execution.

Option	Description
--users[:<option>[=<value>]]*	Query and/or manipulate the user session table. See details below.

User Session Table Manipulation

The following options and values can be used together with the '--users[:<option>[=<value>]]*' command line option:

Option	Value	Description
list	all	List all user entries (default)
	expired	List only expired user entries
remove	all	Remove all user entries
	expired	Remove only expired user entries (default)
	<Session ID>	Remove a user entry by Session ID
time	absolute	Print time values as absolute (default)
	relative	Print time values as relative to now
	internal	Print time values in internal representation
extended	n/a	Use extended column layout (130 chars wide)

Options can be chained and are processed in the order they appear. The options 'time' and 'extended' must appear before the options to which they apply.

The following string abbreviations are permitted:

exp(ired), abs(olute), rel(ative), int(ernal), ext(ended)

Examples:

--users:time=relative:list	List all user entries, relative time
--users:ext:list	List all user entries, extended format
--users:list=exp:remove:list	List and remove expired user entries, then list all (remaining) user entries

Import and Export

Option	Description
-DM<f>	Use Mac dictionary file `<f>` to translate relation names in import file.
-DS<f>	Use server dictionary file `<f>` to translate relation names in import file.

Option	Description
-E<f>	Export the current database to a file named <f>. -E<f> -as <filename> exports to stdout (UNIX only).
-I<f><!>	Import the export file <f> into the current database. If ! is specified, the application version check is ignored. -I<f><!>- as >filename> imports from stdin (UNIX only). Note that the options -UD and -UC are performed automatically when the import is run. First, all instance key constraints in the database to be imported are dropped. After the import, instance key constraints defined in the relation DatabaseRelations are created in the database.
-R<r>	Apply the import/export on the relations specified in <r>, where <r> is a list of relation names delimited by ',. If <r> is preceded by '–', the import/export applies to all relations that are not in the list.
-Q	Create missing database users for view users, who are listed in the database but do not exist in the database. Passwords are set to “123456.”
-T	Tally: Count the number of records and relations in the database. When used with -I, the count is from the specified file to be imported. The database is always left unchanged when -T is specified.
-Z<r>	Zero option.  Use this option with caution, because records will be deleted from the database. -Zt Truncates current relation before import (used with -I). -Zr Requires relation to be empty before importing (used with -I). -ZT Truncates all relations, and terminates.
-3	The server will update three party view group members.

Layouts

Option	Description
-A<dir>	Read Analyze, web and special.txt Analyzer reports from directory <dir>.
-W	The server will import and update the Window and Layout specification from the WindowLayoutList.txt file in the MaconomyDir/ScreenLayouts folder with the layouts found in the MaconomyDir/ScreenLayoutList/Design folder.
-U<x>	Update layouts and manipulate instance key constraints. The argument <x> for this option can be one of the following:

Option	Description
	W — Install and update window layouts on the Standard and Solution layers. (Same as option -W.) P — Install and update print layouts. LP — Install and update localized print layouts. Errors are reported in case of overlapping fields in the layout canvas. L — Install layouts designed using Layout Designer. No MPL dumps will exist for these layouts. Consequently, they cannot be used as a base when defining new MPL layouts. I — Update installation information in database from Dependencies file without the use of a client (faster when installing across a modem connection, for example). A — Update Access Control Information. Also called Enable Security. Updates the database so that users can only see licensed windows, for example, when searching for windows in the client Window Layouts window. This update is also performed the first time that a Windows or Macintosh client runs on a Maconomy system. D — Drops all constraints on instance keys. Deprecated – see -I<f><!>. C — Creates all constraints on instance keys defined in the relation DatabaseRelations. Deprecated – see -I<f><!>. VP — Validate customized print layouts. VW — Validate customized window layouts. EP — Export all customized print layouts to MaconomyDir/ExportedLayouts/. EP:<path> Exports all customized print layouts to <path>. EW — Export all customized window layouts to MaconomyDir/ExportedLayouts/. EW:<path> Exports all customized window layouts to <path>. IP — Import customized print layouts listed in MaconomyDir/ExportedLayouts/ExportedPrintLayouts.txt IP:<path> Imports customized print layouts listed in <path>/ExportedPrintLayouts.txt. IW — Import customized window layouts listed in CustomizationDir/Solution/ScreenLayouts/ExportedWindowLayouts.txt IW:<path> Imports customized window layouts listed in <path>/ExportedWindowLayouts.txt.

Analyzer-Related

Option	Description
--AnalyzerViews	With --AnalyzerViews, the server will install all Analyzer views defined in the XML files in the Analyze folder. The XML files are generated by the MBuilder application.
--CompileAnalyzer <path>	With --CompileAnalyzer, the server will compile Analyzer reports. Use the following command:

Option	Description

	UNIX:
	`macoracle.r.<appshortname> -S<shortname> --CompileAnalyzer <path>`
	
	Windows:
	`MaconomyServer -i<appshortname> -S<shortname> --CompileAnalyzer <path>`
	
	If the path name contains spaces, put it in quotes, for example `MaconomyServer -iW_8_0 -Sw80 --CompileAnalyzer "C:\My AnalyzerReports"`
	
	Any errors from the compilation will be written to the screen.
	
	`<path>` can be a path to a directory or a path to a specific file. If it is a path to a directory, all files with the extension `.gr` in the directory in question will be compiled. Otherwise, only the specific file will be compiled.
	
	The compiled file(s) (with the extension `*.grf`) will be placed in a directory called `CompiledFiles` under the directory in question.

--ConvertAnalyzerFiles	The --ConvertAnalyzerFiles option is a shortcut for calling MBuilder with the option --Analyzer followed by calling the Maconomy server with the option --AnalyzerViews. See the MBuilder manual for details.
--CompileAndConvertAnalyzerFiles <path>	The --CompileAndConvertAnalyzerFiles option is a shortcut for calling the Maconomy server with the options --CompileAnalyzer and --ConvertAnalyzerFiles in succession. First, the Analyzer files in <path> are compiled as described for --CompileAnalyzer <path>. Then, if the compilation succeeded without errors, the compiled *.grf file(s) are copied to the Analyzer folder of the selected application. You are prompted before overwriting existing files in the folder. The compiled files are then converted as described for –ConvertAnalyzerFiles.

M-Script-Related

These options enable you to run M-scripts using the Maconomy server executable instead of the `MaconomyMScript` executable on the web server.

Option	Description
--mscript <filename>	Execute an M-script (in Maconomy server command-line context). This corresponds to executing the script in M-Script stand-alone context, only using the server executable rather than the MaconomyMScript executable. See the M-Script Language Reference manual for details.

Option	Description
--mscriptLogFile <LogfileName>	Specify where M-Script should write its log file. If this option is not used, the log file will be written to `Tmp/MaconomyMScript.log`. However, this setting is subject to being overwritten by the corresponding setting in the `.I` file.

--mscriptIFile <InifileName>	Specify which `.I` file to read when executing the M-script. If this option is not used, the server will attempt to read the `.I` file `MaconomyDir/Definitions/MaconomyMScript.I` if it exists.
	Note that a number of the setup-related `.I` file options that are available to stand-alone M-Script cannot be used in an `.I` file on the Maconomy server. These options include:
	`ServerIP DaemonPort`
	`ListenFrom`
	`Connection`
	`MaxListenPorts`
	`WaitForServer`
	`WaitForDaemon`
	For more information, see the M-Script Language Reference manual.

--mscriptQueryString <QueryString>	Specify a query string to send to the server. This corresponds to the option `-q` for stand-alone MScript. This option can be used multiple times in the same call to the Maconomy server.
--mscriptConfigurationLine <key=value[;key=value]>	Specify a configuration line. This corresponds to adding the line to the bottom of the `.I` file. This option can be used multiple times in the same call to the Maconomy server.

Workflow-Related

Option	Description
--DumpWorkflow template LIFECYCLEDEFINITIONNUMBER [OUTDIR]	This command is used for dumping pre-MWL workflows as a template on the server. `LIFECYCLEDEFINITIONNUMBER` is the definition number of the life cycle. `OUTDIR` is the output folder where the server will dump the MWL workflow template; if not specified, this option defaults to the current working directory. See the Maconomy Workflow Language Reference manual for details.
--InstallWorkflow MWLFILE	This command installs the MWL workflow specified in `MWLFILE` to the custom branch of the server. `MWLFILE` is the MWL specification file. The associated M-script and configuration file must have same base name. That is, if the MWL file is named `timesheetheader.mwl`, the other files must be located in the same folder and named

Option	Description
	`timesheetheader.1.ms` and `timesheetheader.I`. If `timesheetheader.mwl` is located in the subfolder myworkflows relative to the current working directory, it can be installed by issuing the command: `maconomyserver -f --InstallWorkflow myworkflows/timesheetheader.mwl` When the workflow is installed, the Maconomy server renames the installed workflow files to match the workflow name, if they are different. Workflows are installed in the folder `<Appl_home>\MaconomyDir\MScripts\Workflows`. General monitors are installed in the folder `<Appl_home>\MaconomyDir\MScripts\Workflows\General Monitors`. See the Maconomy Workflow Language Reference manual for details.
--VisualizeLifecycle <inputfilename> <outputfilename> [internal|external]	This option tells the Maconomy server to look for the Maconomy Workflow Language file specified as `inputfilename` and produce a visualization of the specified workflow in a file called `outputfilename` and then terminate. The type of the output file is determined by the file extension: if the extension is “gif”, the output format will be `GIF`, and similar for the other formats: `SVG` and `PNML`. The optional third argument specifies which titles should appear on the output graphics: `internal` means that the stages and transitions will be labeled with the value of their `Name` attribute, and `external` means that the `Title` attribute will be used (if it exists).

Other Tools

Option	Description
-FV	Validate all foreign keys, and write errors to standard output.
-FG	Generates SQL commands on standard output, updating foreign key references with invalid spelling (lower case/upper case problems).
--CreateDocumentTables	With `--CreateDocumentTables`, the server will create the tables `TEXTDOCUMENT` and `BINARYDOCUMENT` (with indexes) in the database.
--IP<str>	IP security: Denies or allows server connections based on the remote IP address. This applies to clients, WebDaemons and CGI programs. The options are: `--IPdeny <addresslist>` Add addresses to the list of IP numbers denied access.

Option	Description

	`--IPallow <addresslist>` Add addresses to the list of IP numbers allowed access. where `<addresslist>` has the format: `<address> [ '-' <address> ] { ';' <address> [ '-' <address> ] }`
	
	The `--IPdeny` and `--IPallow` options can be specified in all initialization files. Examples:
	`--IPdeny 10.22.33.44`
	`--IPallow 127.0.0.1; 10.22.33.0 - 10.22.33.255`
	
	The check algorithm is as follows:
	1. Look for address in deny list. If found: Reject connection.
	2. Look for address in allow list. If found: Accept connection.
	3. If not found in either list: Reject connection.
	
	If `--IPdeny` or `--IPallow` are not specified, all connections are allowed.
	
	Note: Remember to quote the options if giving this option on the command line. Example:
	`--IPallow "127.0.0.1; 10.22.33.0 - 10.22.33.255"`

Initialization Files

A number of options are set in initialization files. A total of five initialization files exist.

<appl_ID>.I	Read by MaconomyServer
Maconomy.ini	Read by MaconomyServer
.W	Read by the server when Windows client is launched
.M	Read by the server when Macintosh client is launched
.D	Read by the server when web client (Java client) is launched

The primary server initialization file has the extension `.I`. The prefix (or first name) of the file is usually `MaconomyServer.<appl_ID>`, but another prefix can be specified using the option `-i`. The standard `Maconomy.ini` file is located in the `MaconomyDir/Definitions` folder of the server application to which they apply. Additional files with custom settings can be placed in the folders `CustomizationDir/Custom/Definitions` and `CustomizationDir/Custom.<shortname>/Definitions`,

The server initialization files (`.I` and `.ini`) are read by MaconomyServer after the server has started and executed any command line options.

The client initialization files (`.W`, `.M`, and `.D`) are read by MaconomyServer when a client of a given type is launched. This way, it is possible to specify different options for clients on different platforms.

MaconomyServer.<appl_ID>.I

The following options can be entered in the initialization file for the Maconomy server.

Server Operation

Option	Description
-b	Enable server batch print mode. See also the Product Note “Maconomy Batch Printing”.
-c<n>	Use CC_Table <n> as default. A CC_Table performs character conversion between platforms.
-d	Activate debug output. Debug output can be relevant when resolving problems. However, activating debug output can hit server performance.
-D<x>	When the debug file reaches <x> bytes, close it and open a new one with the same name (overwriting the old one) to save disk space. Debug files are created in the MaconomyDir/Tmp directory and named in the following way: <x>Debug<y..y>, where <x> can have the following values: O When running an Oracle™ database. 2 When running an IBM DB2™ database. M When running an Microsoft SQL Server™ database. X When running MaconomyServer with the -x<p> option I When importing a database into Maconomy with the -I<f><!> option. The <y..y> part of the debug filename is a random name. The length varies according to platform.
-e<v>	Sets an environment variable. <v> has the form VAR=VALUE. You can, for example, use this to specify Oracle and Maconomy home or the current language. Examples: -eMACONOMY_HOME=F:\MaconomyNT\Maconomy\w_8_0 -eORACLE_HOME=d:\oracle\ora81 -eORACLE_SID=ORCL -eNLS_LANG=AMERICAN.WE8ISO8859P1
-r<n>	Set record container cache size. Limits the number of lines (<n>) that can be displayed in one dialog in the Java client and in a dialogGet executed in M-Script. The default value is 5000. Increasing this value will increase memory usage by 16 bytes * <n> * the number of web daemons or Maconomy servers. Increasing the value significantly will also increase CPU usage.
-s<c>	Oracle only: Use <c> as SQLNET connect string (in 3-tier setups with the database on another host).
-t	NT only: Enable stack trace which can be used for debugging on the Windows platform.
-P	Sybase only: Use stored procedures instead of direct executes. Deprecated; do not use.
-CS	Show the server’s current configuration settings.

Web Operation

Option	Description
-w<x>	The server will wait <x> milliseconds before timing out when communicating with a CGI client in web mode. The default value is 30000 milliseconds (30 seconds). Can be set in the .D initialization file as well.
-C<x>	The server will cache <x> Analyzer layouts during web execution. The default value is 5. Can be set in the .D initialization file as well.

Options Both on Command Line and in Initialization Files

The following options can both be set on the command line and in the server initialization file. Please refer to the preceding descriptions for the following options:

```
-f
--AnalyzerViews
--CreateDocumentTables
--IP<str>
-A<dir>
```

Maconomy.ini

The settings in this file are read by the Maconomy server. The file is reset to default settings every time that you upgrade Maconomy. The file has the format of ordinary Windows initialization files with section headings in the form `[section]` and entries in the form `key=value`.

To add or change settings which should be preserved during upgrades, you can create a new 'Maconomy.ini' file in the Maconomy server's customization folder hierarchy – either in 'CustomizationDir/Custom/Definitions' or ' CustomizationDir/Custom.<shortname>/Definitions' (for shortname-specific configuration settings). Settings read from these files will take priority over any settings in the standard 'Maconomy.ini' file.

Settings in ServerConfig

The following settings can be specified in the section [ServerConfig]. The value type LIST means that you can specify one or more of the following values: WebClient, WinClient, MacClient, ALL, and NONE, separated by a comma, if you specify more than one.

Name	Type	Description
_NoRestrictionOptimization	None	This key does not have a value – Maconomy simply checks for the existence of the key.
AllowNonSSLClients	LIST	If you need to run the Macintosh client in an SSL environment, you must set this key to the value MacClient.
AllowOldClients	LIST	
AllowWinClientSansXE	BOOLEAN	This is the language used when localizing set-up data. Example: W

Name	Type	Description
DefaultEncoding	STRING	In this section, specify a character encoding scheme to be used by the Maconomy client or the Java™ platform. Specify the character set used by the server.
DefaultReportEncoding	STRING	Default user language in the clients. If this is not set, the setting for EnterpriseLanguage is used. Example: W_MCS
DisableSSL	LIST	Specify whether Secure Sockets Layer (SSL) should be activated for the server when using the client(s) specified in LIST.
IPAddress	STRING	
LifeCycleCheckHighLevelLock	BOOLEAN	If this setting is true, Life Cycle Actions are only enabled if no other user holds a high level lock on the current record.
LogAllUpdates	BOOLEAN	Enables logging of all updates in the application for relations defined in the file UpdateLog.cnf. For more information, see the chapter "Advanced Logging". Important note: This setting is experimental and may cause duplicate entries in the update log.
ROCacheActivated	BOOLEAN	If the value is true (default as of 9.0 SP 5), Maconomy will cache all universe and Analyzer reports run by users, according to the specifications set by the other RO* options specified in the file. The caching system prevents the recalculation of universe and Analyzer reports in the Portal and in the Java client when a report is printed. It is recommended to activate the caching system, as doing so may lead to considerable performance improvements. If the value is false, the other RO* options do not apply.

Name	Type	Description
ROCacheNumOfRowsMAX	LONG	Controls the number of Report Objects to be cached. The default value is 100 objects. When the number of objects reaches this value, the oldest object will be "flushed" (deleted) from the cache
ROCacheTimeoutMIN	LONG	The Report Object timeout value in minutes. The default value is 10 minutes. After this time, the report object is deleted.
ROMaxNumberOfRows	LONG	Set the maximum number of table rows to process in MPL1-3 prints.
ROMaxNumberOfRowsStrict	LONG	Like ROMaxNumberOfRows except the print will fail if the limit is exceeded.
ScriptCustomizationDisabled	BOOLEAN	Specifies if string comparisons are case sensitive. String comparisons were made case sensitive as of version 7.1. This option can be set to false by M-Config if you are using a case-insensitive SQL Server database.
SingleLogin	LIST	Specify whether Single Sign On (SSO) should be activated for the server when using the client(s) specified in LIST. Valid members of LIST are WebClient, WinClient, All, and None. See the chapter "Single Sign On with Kerberos."
SSO_ExternalVerifier	STRING	SSO: The path to the external verifier (Kerberos). See the chapter "Single Sign On with Kerberos.".
SSO_KerberosKDC	STRING	SSO: The network name or IP address of the Kerberos KDC (Key Distribution Center). See the chapter "Single Sign On with Kerberos.".

Name	Type	Description
SSO_KerberosRealm	STRING	SSO: The name of the realm (domain) under which the Maconomy service has been registered. See the chapter "Single Sign On with Kerberos."
SSO_ServiceName	STRING	SSO: This is the name of the service Maconomy has been registered as at the Single Sign On authenticator. See the chapter "Single Sign On with Kerberos.".
NativeSingleLogin	BOOLEAN	The support for native SSO on Windows when logging in using the Java Web Start client can be disabled by setting the property: "NativeSingleLogin" to false in the Maconomy.ini
SSO_KerberosRealms	STRING	For multi-realm authentication to work, this property must be set to contain a list of all the realms from where SSO should be possible, along with either the network name or the IP address of the associated KDC. <realm1>=<kdc1>,<realm2>=<kdc2>,...
SSO_AllowWeakCrypto	BOOLEAN	If, when upgrading Java to version 8, issues should arise because of old systems still using DES this new option has been added. The default value is "false". If set to "true" the Java Client and the Maconomy Portal will allow the use of DES as a supported encryption type when handling Kerberos Tickets.
RPC_UserIdleTimeout	INTEGER	The timeout of idle users connected via the RPC protocol. This includes the Workspace Client and REST API.
RPC_Log = true|false	BOOLEAN	The collected information will

Name	Type	Description
		be written to log files beneath the specified folder path, or to the default temporary directory (for example, 'X:\Maconomy\Tmp' or '/tmp'), with the following naming scheme: RPC_Log-<PID>-<TIMESTAMP>.log
`RPC_LogFolder =`	STRING	<folder-path>
`RPC_LogTruncateOn`	ENUM	login|reconnect|detach|logout|all

Settings in ApplicationConfig

The following settings can be specified in the section `[ApplicationConfig]`.

Name	Type	Description
`ApplicationVersion`	STRING	The Maconomy version running on the current server
`DefaultSimpleSearchTab`	INTEGER	Specifies if “simple search” should be default when pressing CTRL+F in the Windows client
`EnterpriseLanguage`	STRING	This is the language used when localizing setup data. Example: W
`LifeCycleCache`	BOOLEAN	The Maconomy server by default caches life-cycle (workflow) scripts (such as custom action guard scripts). However, during development of such scripts, it is beneficial to set this option to false so that you do not have to restart the Maconomy server every time you change the script.
`MSLCaseSensitive`	BOOLEAN	Specifies if string comparisons are case sensitive. String comparisons were made case sensitive as of version 7.1. This option can be set to false by M-Config if you are using a case-insensitive SQL Server database.
`MPMLogDirectory`	STRING	The path specified here indicates where the Maconomy Performance Monitor output log is placed. If this option is not present, or if the folder is not accessible for some reason, the MPM functionality is turned off completely. See the chapter “Maconomy Performance Monitor.”
`PPUSQLAccess`	ENUM	ENUM can have the value write (default), read, or off. This makes it possible to grant full access to modify the contents of the database using PPUSQL. Full access is only granted if certain other criteria are met (see the PPUSQL manual for more details). If set to read, the contents of the database may only be read with PPUSQL. Off completely prohibits PPUSQL from connecting to the database.

Name	Type	Description
ReportLeftMargin	LONG	Creates additional left margin in RGL printouts. The extra margin is specified in points.
ReportTopMargin	LONG	Creates additional top margin in RGL printouts. The extra margin is specified in points.
SSO	ENUM	ENUM can have the value off (default), namematch, or namemapping. Specifies if Single Sign On is enabled, and the authentication method used if enabled. See the chapter "Single Sign On with Kerberos".
SSODomain	STRING	List of domain names. If Single Sign On (the old version without external verifier) is enabled, the user must exist in a domain specified here. See the chapter "Single Sign On."
SSOAllow	STRING	See SSODeny.
SSODeny	STRING	Together with SSOAllow this option defines a range of valid IP addresses when using Single Sign On (the old version without external verifier). See "Single Sign On."
TechMissingStageMessage	BOOLEAN	You can choose to let the workflow engine issue very technical error messages or more general messages. If this setting is false or undefined, a message might look like this: "The Life Cycle Definition '^0' for the object does not allow the operation to be executed. (Transition '^1' cannot be executed as Stage '^2' is not active)". This is useful while debugging workflows. If this setting is true, the same message would be something like "The operation cannot be completed as it is in conflict with the workflow"
UserLanguage	STRING	Default user language in the clients. If this is not set, the setting for

Name	Type	Description
		EnterpriseLanguage is used. Example: W_MCS
ExportAccessControl	BOOLEAN	Controls whether access to "Export..." in the Groups window also controls access to "File -> Export Table.." in the Java client. The default is true to allow access control table and search result data exports.

Settings in JavaClientConfig

The settings in the section `[JavaClientConfig]` concern the functionality of the Maconomy client for the Java™ platform.

Name	Type	Description
DefaultEncoding	STRING	A similar key in the section [ServerConfig] specifies the default character encoding scheme for Maconomy clients. Use this setting to specify a character encoding to be used by the Maconomy client for the Java™ platform, if the name of the encoding is different on the Java platform. For more information, see “Maconomy Client for the Java™ Platform in the document “Character Conversion.”
DefaultFont	STRING	Change the default font used by the Maconomy client for the Java™ platform. You can for instance choose to do this if the default font (Arial) is not supported by the selected character encoding as specified above. For more information, see “Maconomy Client for the Java™ Platform Setup” in the document “Character Conversion.”
ExitJavaVMOnLogout	BOOLEAN	The error “Cannot Initialize Java Applet" is displayed when you are logging in or out of Maconomy Portal while selecting Java Applet-based Maconomy Portal components. The fix for this is default enabled (true) but it can be turned off setting the parameter to false.
RecreateDesktopIcon	BOOLEAN	On Windows, sometimes the

Name	Type	Description
		creation of the desktop shortcut of the Java client fails. This is caused by a bug in Java Web Start. To work around this, has been added. The default is true.
UseOKAsDefaultInWarnings	BOOLEAN	It is possible to change the behavior of <enter> in the Java client and Java applets so that <enter> selects "OK" in warning dialogs. The default is true.
WaitingForServerEnabled	BOOLEAN	Controls whether the "Waiting for data from server..." progress dialog is shown at all. If this setting is set to "false," the mechanism is suppressed completely. The default is true.
WaitingForServerRepaintEnabled	BOOLEAN	Controls whether a "repaint" of the GUI precedes the "Waiting for data from server..." progress dialog. This can be used as a less disruptive way of updating the client GUI when waiting for a long running server request. This feature is disabled by default. The default is false.
WaitingForServerRepaintDelay	INTEGER	Controls the delay before the client GUI is repainted when waiting for a long running server request. See also "WaitingForServerRepaintEnabled." The default is 500.
WaitingForServerDialogDelay	INTEGER	Controls the delay before the "Waiting for data from server..." progress dialog is shown. See also "WaitingForServerDialogDelay." The default is 2000.
RefreshDialogsAfterErrorsEnabled	BOOLEAN	Controls whether all open tabs / windows are refreshed after server / network problems was detected by the Java client. This ensures that all tabs / windows pick up any changes made by the server before it crashed. The default is true.
BlockOnExecuteROEEnab	BOOLEAN	Controls whether server requests related to the execution of the

Name	Type	Description
led		Analyzer waits until other server requests have finished (and vice versa). This ensures that only one server request is executed at any time. If this setting is set to "false", then the mechanism is suppressed completely and requests related to the execution of the Analyzer is executed immediately without waiting for other requests to finish. The default is true.
BlockOnExecuteNotificationEnabled	BOOLEAN	Controls whether server requests related to the update of notifications waits until other server requests have finished (and vice versa). This ensures that only one server request is executed at any time. If this setting is set to "false", then the mechanism is suppressed completely and requests related to the update of notifications is executed immediately without waiting for other requests to finish. The default is true.
OptimizeGetFavoritesEnabled	BOOLEAN	The default value is true. When this feature is enabled, the set of M-Script functions called when favorites are calculated is the same for the initial set of favorites when dialogs are opened, but subsequent calculations of favorites will only call the M-Script functions particular to the favorite being updated whereas previously the set of M-Script functions called was the same the first time and subsequent time. If your favorite M-Scripts rely on the old behavior (for initialization of global data), this feature either needs to be turned off or the M-Scripts need to be changed to take this into consideration.
DebugLog_LogLoginTimeoutLoggingEnabled	BOOLEAN	This option control if, during login, a debug log message will be sent to the web server if the login process takes longer than the number of

Name	Type	Description
		seconds Specified – see DebugLog_LogLoginTimeout
DebugLog_LogLoginTimeout	INTEGER	Number of seconds
DebugLog_SeparateJVMLoggingEnabled	BOOLEAN	This option makes it possible to control whether a debug log will be sent to the web server if a Java applet detects that it has been started in a JVM separate from the JVM running the Login Applet. Default is false
DebugLog_AskUserEnabled	BOOLEAN	It is possible to show a yes/no alert to the user before sending the debug log to the web server. The default is false. This option only have any effect if sending back debug logs to the web server is enabled.
DebugLog_AskUserMessage	STRING	It is possible to show a yes/no alert to the user before sending the debug log to the web server. The default is "". This option only have any effect if sending back debug logs to the web server is enabled.
NavigateUsingFindSortOrderEnabled	BOOLEAN	The unnecessary alert "This is the first Group in the search pane" appears when you navigate down in search results. This can be turned off. The default is false.
WindowsScreenMenuBarWorkAroundEnabled	BOOLEAN	This is the same issue as MacOSXScreenMenuBarWorkAroundEnabled, but for Windows. If it should appear, the default is false.
MacOSXScreenMenuBarWorkAroundEnabled	BOOLEAN	The screen menu bar items do not function correctly when running under Java 1.7u25 on Mac OS X, a workaround is active to fix this. The default is true.
FindNavigationDelay	INTEGER	Defines the delay before navigation in the "Find" pane takes effect, the default is 500ms to match the previous behavior of the Java Web

Name	Type	Description
		Start client and Java Applet's.
AutopilotNavigationDelay	INTEGER	Defines the delay before Autopilot navigation takes effect, the default is 2000ms to match the previous behavior of the Java Web Start client and Java Applets.
GlobalAutopilotNavigationOnDialogActivateEnabled	BOOLEAN	Defines whether Autopilot navigation should take effect immediately when a tab / window is activated. This is convenient in that the contents of the tab / window will always reflect latest Autopilot navigation when activated, the default is "true" to match the previous behavior of the Java Web Start client and Java Applets.
DependencyUpdateDelay	INTEGER	Setting this to something other than zero will delay the update of tabs / windows by the specified amount of ms. The default is 0 to match the previous behavior of the Java Web Start client and Java Applets.
GlobalDependencyUpdateOnDialogActivateEnabled	BOOLEAN	Defines whether the update of *all* tabs / windows whose contents might be changed by a change in another tab / window should take effect immediately when *any* tab / window is activated. This is convenient in that the contents of the tab / window will always be up to date, the default is "false" to match the previous behavior of the Java Web Start client and Java Applets. If setting "DependencyUpdateDelay" to something else than 0, you might consider enabling this setting.
LocalDependencyUpdateOnDialogActivateEnabled	BOOLEAN	Defines whether the update of a tab / window whose contents might be changed by a change in another tab / window should take effect immediately when the tab / window is activated. This is convenient in that the contents of the tab / window will always be up to date, the default is "false" to match the previous behavior of the Java Web

Name	Type	Description
		Start client and Java Applets. If setting "DependencyUpdateDelay" to something else than 0, you might consider enabling this setting. Note the difference to the "GlobalDependencyUpdateOnDialogActivateEnabled" setting, that setting, if enabled, will update *all* tabs / windows if needed. The "LocalDependencyUpdateOnDialogActivateEnabled" will only update the activated tab / window, which can be much more efficient. It does not make sense to enable both of these settings.
`DependencyUpdateOnFavoritePopupActivateEnabled`	BOOLEAN	Defines whether the update of a tab / window whose contents might be changed by a change in another tab / window should take effect immediately when a favorite popup is activated. This is convenient in that the contents of the favorite popup will then be calculated using data that is always up to date, the default is "false" to match the previous behavior of the Java Web Start client and Java Applets.
`RefreshOnLowerAutopilotNavigationEnabled`	BOOLEAN	Is by default "true," which corresponds to the behavior of the Java Web Start client and Java Applets in previous versions of Maconomy. If this option is set to "false," navigation between table rows resulting in navigation in other tabs / windows will no longer force a refresh of those tabs / windows.
`WorkAreaBackupIntervalEnabled`	BOOLEAN	Callback requests from the server do not block other requests until callback has been finalized. This setting control if work area user settings are backed up in the background. The default is true.
`WorkAreaBackupInterval`	INTEGER	This setting control how often the work area user settings are backed up if enabled. The default is 300 seconds.

Name	Type	Description
`ClientLineBreaksEnabled`	BOOLEAN	Controls whether the feature is enabled. If ”false", the feature is not functional and the "Use client line breaks" check box in the "Preferences" window is hidden. If "true" the feature is functional and the "Use client line breaks" check box in the "Preferences" window is shown.
`UseClientLineBreaksDefault`	BOOLEAN	Controls the default value of the "Use client line breaks" checkbox in the "Preferences" window. If this setting is not specified, the "Use client line breaks" check box is left unchecked if not already checked by the user.

Settings in WindowConfig

The settings in the section `[WindowConfig]` concern the functionality of the Maconomy clients:

Name	Type	Description
`JumpToLast`	STRING	Forced the specified dialog to open on the last record instead of the first. Specified like this: TimeSheets=JumpToLast
`NotificationsEnabled`	BOOLEAN	Whether notifications should be displayed for outstanding To-dos and alerts. By default this setting is “true.”
`InitialNotificationInterval`	INTEGER	The interval (in seconds) from the client is started until it will begin to check the Maconomy server for new notifications. The default value is 0 seconds.
`NotificationInterval`	INTEGER	The interval (in seconds) with which the client will check the Maconomy server for new notifications. The default value is 300 seconds (5 minutes).
`IgnoreVersionChecks`	BOOLEAN	If true Java client and Applets ignore version checks during startup.
`RichText`	BOOLEAN	To control whether text fields should appear as "richtext" fields from MDL

Name	Type	Description
		The default is true.

Deprecated Settings

Some settings are used for temporarily bridging a gap between two supported versions of Maconomy when new technology is introduced. Over time, these are deprecated, and they may be removed in later versions. Deprecated settings include (all in the section `[ApplicationConfig]`):

Name	Type	Description
`AutoUpdateClient`	BOOLEAN	Deprecated. With this option, you can overwrite the standard configuration setting which by default tells Maconomy to overwrite (update) the file DBDesc on the client machine. There is no reason to set this value to false.
`ExceptionIfDiffWithCaseSensitiveCompare`	BOOLEAN	Deprecated. This was used in the transition from case insensitive to case sensitive string comparison in version 7.1. See the setting MSLCaseSensitive.
`HLLVersion`	INTEGER	Deprecated. The high level lock version is always 2.
`KeepLocalizationCommentInMDL`	BOOLEAN	Deprecated. All prints are based on MPL.
`LifeCycles`	INTEGER	Deprecated. The Maconomy sets the lifecycle version level to 5 by checking a database table.
`LocalizeMSL`	BOOLEAN	Deprecated. MSL constants are always localized.
`LongPasswordCrypt`	BOOLEAN	Deprecated. This was introduced in a Service Pack for version 8.0 and specifies if all characters in a password should be used in the generation of the crypt code for a password. This became the default as of version 9.0.
`MaconomyWorkBench`	BOOLEAN	Deprecated
`PatchLevel`	STRING	Deprecated. With this option you can specify the Service Pack level of the current application. This information is now located in the version.info file, which only Maconomy can change.

Name	Type	Description
RecordAccessControl	INTEGER	Deprecated. The record access control version is always 2.
RunTimePrinterDependence	BOOLEAN	Deprecated. Should MPL prints be formatted with respect to the current printer or using "standard" margins etc. Always true.
UseMPLExclusively	BOOLEAN	Deprecated. All prints are based on MPL.
UseODLDefaultForEnum	BOOLEAN	Deprecated. Maconomy always uses the ODL default for popup values.
ValidateForeignKeys	BOOLEAN	Deprecated. Foreign keys are always validated.

MaconomyServer.<appl_ID>.W

The settings in this file apply to the Windows client and are read by the server when a Windows client is launched. The following options can be entered.

Option	Description
-pC	For internal use. Is present by default.
-c<n>	Use CC_Table `<n>` for Windows clients. A CC_Table performs character conversion between platforms. This table should normally be `CC_Table.Win`, but may be different in countries using non- standard character sets. Note that a default CC_Table can be specified on the command line as well.
-e	Add environment variable to apply to Windows clients. For more information, please see the description of the `-e<v>` parameter above.

MaconomyServer.<appl_ID>.M

The settings in this file apply to the Macintosh client and are read by the server when a Windows client is launched. The following options can be entered.

Option	Description
-pC	For internal use. Is present by default.
-e	Add environment variable to apply to Macintosh clients. For more information, see the description of the `-e<v>` parameter.

MaconomyServer.<appl_ID>.D

The settings in this file apply to the Web client (“D” stands for the “(Web) Daemon” CGI program handling Web client requests) and are read by the server when a Windows client is launched. The following options can be entered.

Option	Description
-w<x>	The server will wait `<x>` milliseconds before timing out when communicating with the Displayer program or the Web Daemon. The default value is 30000 milliseconds (30 seconds). Note that this option can be specified in the server `.I` file as well.
-C<x>	The server will cache `<x>` Analyzer layouts during Web execution. The server keeps a cache of certain database information (SQL cursors, and so on) related to an Analyzer report. By caching this information, the server can access the database faster (however, data is read from the database every time). Default value: 5. Note that this option can be specified in the server `.I` file as well.
-e	Add environment variable to apply to the Web Daemon. For more information, please see the description of the `-e<v>` parameter above.

International Settings

The Maconomy server is by default configured to use Danish settings for date and number formats. This can be changed in the Maconomy client, but when running reports directly on the server, for example, you need to customize the server to use the date and number formats that apply at your location.

The date and number configuration is stored in the environment variable `INTL_SETUP`. For information about changing this variable, please refer to the documentation of the operating system of your server, or see the description of the initialization file parameter `-e<v>` .

Default Settings

By default, the environment variable `INTL_SETUP` has the following value:

`/DMY110:010000C,.0011,00011·A····A····`

a total of 38 digits, where “·” indicates a blank value (a space).

The following table specifies the meaning of each digit, starting with digit “0.” The value “1” means “true.” Alphabetic characters (“A”-”Z”) are converted to decimal values, where “A” equals “0,” “B” equals “1,” and so forth. Hence, if you want the value “2” in digit 14 (“No. of decimals for integers”), you should enter “C,” because “C” equals “2.“ This is to avoid having to enter the ambiguous number “0.”

Digit	Pertains to	Function	Default
0	Date functions	Date separator	-
1		Format - sequence of day, month, year	D
2			M
3			Y
4		Leading zero, day	1

Digit	Pertains to	Function	Default
5		Leading zero, month	1
6		4-digit year	0
7	Time functions	Time separator	:
8		Leading zero, hour	1
9		Include seconds	1
10		12-hour clock (AM/PM)	0
11	Integers	Thousands separator	0 (none)
12		Parenthesis around negative numbers	0
13		Trailing minus	0
14	Amounts	No. of decimals	C
15		Decimal separator	,
16		Thousands separator	.
17		Parenthesis around negative numbers	0
18		Trailing minus	1
19		Trailing decimal zeros	1
20		Leading integer zero[a]	1
21	Reals	Decimal separator	,
22		Thousands separator	0 (none)
23		Parenthesis around negative numbers	0
24		Trailing minus	0
25		Trailing decimal zeros	1
26		Leading integer zero[a]	1
27		No. of decimals	<blank>[b]
28	Time Strings	Length of string for “Morning” (AM)	A[c]
29		“Morning” character 1	<blank>
30		“Morning” character 2	<blank>
31		“Morning” character 3	<blank>
32		“Morning” character 4	<blank>
33		Length of string for “Evening” (PM)	A[c]
34		“Evening” character 1	<blank>
35		“Evening” character 2	<blank>
36		“Evening” character 3	<blank>
37		“Evening” character 4	<blank>

a. Zero before decimal if number is less than 1

b. <*blank*> specifies the default number of decimals, which is 2. This could also be written as "C"

c. "A" = 0 - AM/PM is not entered. Remember to set digit 10 = 1. Max. number of characters is "E" (four characters).

Example

For example, if you want to run a report `myReport.grn` and you want the real numbers in your output to be displayed with five decimals, perform the following steps (example from Windows NT).

1. Change the international settings with the following command:

   ```
   set INTL_SETUP=/DMY110:010000C,.0011,00011FA···A
   ```

   "·" signifies a blank (space). Note that it is not necessary to enter all of the digits, only those up to and including the digit whose value you want to change. For instance, in the preceding example, the last 4 blank values have been left out.

2. Run the report using the –x option:

   ```
   maconomyserver -i<inifile> -xmyReport.grn
   ```

For more information about running reports on the server, see "Product Note: Running Programs and Reports on the Server."

Shortcut to the INTL_SETUP Variable

If you have set up your system to support batch printing, you can retrieve a properly formatted `INTL_SETUP` from an option file created by the batch printing system.

When a Maconomy client sends a print job to the server, an option file is created. The second line in this file is the date and number format string, which is formatted according to the above specifications. However, the string does not contain the default value of the Maconomy server; it contains the setting of the client that issued the print job.

Hence, you can set up a client (using the window Preferences) to use the formats you want in the report, send a batch print job to the server, and then take the format string from the batch job option file on the server and use it as input to the `SET INTL_SETUP` command mentioned above.

For more information about batch printing, see "Batch Printing" in this manual.

CheckOracle

The `CheckOracle` utility is used to check the condition of the Maconomy database.

To run `CheckOracle`, complete the following steps:

1. Open a command prompt/shell.
2. Run the following command:

   ```
   CheckOracle system/manager [-a]
   ```

If you are running Windows, and the command is not recognized, change to the `<MaconomyHome>\bin` directory before running the command.

The `-a` option produces a more detailed report. The CheckOracle utility is described in detail in "Check Oracle" in "General Server Maintenance" in this manual.

Utility Reference

This section lists all of the tools and utilities in the Maconomy `/bin` directory. It is intended as a short overview of what each tool does. Some tools are described further in this document or elsewhere in this manual; whenever that is the case, a reference to the section in question is made.

Note that a number of the utilities are only used when installing or updating Maconomy. This is only done by Maconomy consultants and partners, and the utility is, therefore, only described very briefly.

Executable Files

The following is a description of each executable file in the Maconomy `/bin` directory. The tools are listed in alphabetical order.

Name of Executable	Description
binsum.exe	Help utility for checking the contents of files that have been transferred from a UNIX machine to see if the transfer was successful.
ByteSwapResourceFile.exe	For internal use.
ConvertTextFile.exe	Used by switch CC (Character Conversion) files between UNIX and Macintosh formats. Usage: `ConvertTextFile [-recurse] [-cctable <path>] [-v] [-r] [-n] [-l] <Filename>*` `-recurse`: all subdirectories are searched using * `-cctable <path>` : Path to CC tables (`CC_Table` and `CC_Table.MacToServer`) `-v`: Print version info and exit `-n`: Normal conversion (FROM Mac) - default `-r`: Reverse conversion (TO Mac) `-l`: No conversion (change line feeds only) `-h`: This help
DBLocalize.exe	For internal use.
Display2.exe	This is the program that displays the “old” Maconomy client (version 2) for the Java™ platform. The executable is placed in the web server `cgi-bin` directory if used.
EditStringNrResource.exe	For Maconomy consultants only.
EditStringResource.exe	For Maconomy consultants only.
Jobserver.exe	The Maconomy executable that controls batch jobs on the server. For more information, see “Introduction” in the chapter “Batch Printing.”

Name of Executable	Description
MaconomyDaemon.exe	The Maconomy service. For more information, see “Maconomy Server Options and Parameters” in this manual.
MaconomyMScript.exe	The Maconomy M-Script executable. Should be placed in the web server `cgi-bin` directory, and can be renamed.
MaconomyPrinterDriver.exe	A Maconomy printer driver, which outputs reports to printers. For more information, see “MaconomyPrinterDriver” in the chapter “Batch Printing” in this manual.
MaconomyServer.exe	The Maconomy server executable for Oracle databases. Can be run to perform server functions, such as validating print layouts, and so forth. For more information, see “Maconomy Server Options and Parameters” in this manual.
MaconomyServerMSQL.exe	The Maconomy server executable for MS SQL Server databases. Can be run to perform server functions, such as validating print layouts, and so forth. For more information, see “Maconomy Server Options and Parameters” in this manual.
MBuilder.exe	Used for installing Universes (MUL files), Reports (MRL files), Report Layouts (MPL files), and external relations to a Maconomy installation (MOL files) on the Maconomy server. For more information, see the MBuilder Reference.
MDumper.exe	A Maconomy installation contains a huge amount of information. To gain access to some of this information, extracts must be made in specific formats to use it properly. MDumper is the tool to extract, or dump, information from a Maconomy installation. The information dumped by MDumper is used in various independent ways. Some of the installation information is valuable knowledge when developing customized extensions to Maconomy, such as Universes and Universe Reports. To support easy access to this information, MDumper features the concept of MDoc. MDoc is a tool for generating a set of hyperlinked HTML files with which you can visually browse through the basic structures of the Maconomy installation. For more information, see the MDumper Reference.
mkconf.exe	For internal use.
ModifyExportFile.exe	Used for removing data from an export file, leaving only setup data. For use by Maconomy consultants only.

Name of Executable	Description
MPLDumper.exe	Used for dumping old-style (pre-MPL) print layouts (created using the Layout Designer) to a text file.
MStamper.exe	A Maconomy solution can contain a large number of data, format, and script files that are used for reporting, integration, and web-based presentation. The authenticity and integrity of all of these files is paramount to ensure smooth operations, and information concerning who wrote what is essential for directing questions and support calls to the right parties. Maconomy solves these issues by providing all such files with a stamp that protects the files from accidental corruption or malicious tampering. The stamp also contains essential information about the origin of the file, such as author and creation date, and can also be used to restrict the distribution of the file in various ways. The MStamper tool enables external consultants and Maconomy partner organizations to stamp their products as they see fit, thereby getting the same high level of control over the authenticity and distribution of their work. For more information, see the MStamper Reference.
precompile.exe	Used for generating platform-specific installation scripts from a cross-platform foundation. For use by Maconomy consultants only in connection with installing/upgrading Maconomy.
PrintToPDF.exe	A Maconomy printer driver, which outputs reports in PDF format. For more information, see “PDF Printing” in the chapter “Font Administration in Maconomy.”
PWChange.exe	For use by Maconomy consultants only. Used for generating clear-text passwords in connection with changing passwords on servers with extended security, that is, where the database user password differs from the maconomy user password. Used with Oracle databases.
PWChangeMSQL.exe	Same as above, but for MS SQL Server databases.
Sleep.exe	For internal use.
TAR.EXE	Used for unpacking UNIX-style “tarballs”—files that contain a number of other files, which may or may not be compressed. For instance, the Maconomy DPU consists of a tarball that contains the online reference manual in HTML format.
ViewGenerator.exe	For use by Maconomy consultants only. Used for generating the database views through which Maconomy users view data.

Name of Executable	Description
vttool.exe	For internal use.
WebDaemon.exe	The program that establishes communication between web clients and the Maconomy server. For more information, see the chapters “System Overview” and “Web Daemon Options and Parameters.”
Xalan.exe	Used by MaconomyMScript.exe when generating or parsing XML files for use in the Portal.

Script Files

The following is a description of each command/batch file (.cmd) in the Maconomy /bin directory.

Note that many of the scripts listed are for use by *Maconomy consultants only*.

They are listed in alphabetical order.

Name of Command File	Description
CheckOracle.cmd	Used for checking the status of the Oracle database.
Delete3PStuff.cmd	Script for deleting third-party stuff. Used in connection with upgrades.
DeleteAnalyzerViews.cmd	Script for deleting Analyzer views. Used in connection with upgrades.
DeleteDHViews.cmd	Script for deleting DialogHandler views when updating the application version.
DropPortalConstraints.cmd	Script for dropping constraints on portal tables. Used in connection with upgrades.
oraeater.cmd	This tool reveals the most “resource-eating” Oracle threads for the last 20 seconds. It is used for fault-finding. Usage: `oraeater <username> <password> [-a]` The -a option shows the most resource eating Oracle threads since Oracle was last launched. The current thread and the Oracle system threads are not shown. To kill an Oracle thread, use `orakill <ORACLE SID> <threadno>`
Recreate3PStuff.cmd	Used in connection with upgrades.
UpdateFieldsAndRela-tions.w_8_0.cmd	Used in connection with upgrades.

Coupling Service

The Coupling Service is a server-side component that was introduced with the Maconomy 2.0 Workspace Client. It is a part of the Maconomy server, but it also acts as a gateway between the Client and the Maconomy server daemon—either the Windows Maconomy Daemon or the UNIX internet daemon (inetd/xinetd).

The Coupling Service contains some server functions, but it also fulfills much the same role for the Maconomy 2.0 Client as the WebDaemon and Jaconomy executable do for the existing Java Client. It is written in Java and built on top of the Eclipse Equinox implementation of the OSGi platform framework.

System Requirements

The following general system requirements apply specifically to the Coupling Service:

- 4 GB free memory + 1GB free memory / 1000 users
- 500 MB free disk space
- Server OS: Microsoft Windows Server (64-bit), RHEL (64-bit), AIX (64-bit), and Solaris Sparc (32-bit)

For example: a standalone Coupling Service configured for 2000 users should have at least 6GB of free memory. The Maconomy Application server should be sized with similar free memory, and a combined server for both the Maconomy Application server and the Coupling Service for 2000 users should have at least 12GB of free memory.

Refer to the most recent version of the *Maconomy Supported Products* document for more details about the exact systems and versions that are currently supported.

System Architecture

The Maconomy server consists of both the Maconomy Application Server and the Coupling Service. In addition to being part of the server, the Coupling Service functions as a gateway for the Workspace Client to the Maconomy Application Server. The service must be able to connect to the Maconomy server on the port number that is offered by the server daemon, and to expose its own service ports where clients can connect.

The Coupling Service often needs to communicate extensively with the Maconomy Application Server, which means that parameters such as bandwidth and network latency between the Coupling Service and the server are critical.

The following sections briefly discuss a few possible setups and their impacts on system performance.

Same Server

The recommended scenario has the Coupling Service installed on the same physical hardware as the Maconomy Application Server, such that communication between the processes can happen without needing to go through external network interfaces.

This setup should provide the best performance possible.

Co-Located Server

The Coupling Service can be installed on a separate server machine. Such a setup might be appropriate if the Maconomy Application Server is already under a high load, or if network architecture considerations make it undesirable to open the required ingoing ports on the server machine.

Provided that the two servers are co-located and connected by a high-performance (Gigabit) network, this setup can generally be expected to provide an acceptable level of performance.

Remote Server

In this scenario the Coupling Service is installed on its own server at a remote location. This setup makes the Coupling Server vulnerable to the exact network topography between the two server locations.

Limitations in the bandwidth (for example, a 10/100 MBit Ethernet) or even moderate network latencies could cause an unacceptable degradation in performance, and therefore this setup is ***not* recommended**.

Installation and Configuration

This section describes the basic steps involved in installing and configuring the Coupling Service.

MConfig is able to install and provide basic configuration of the Coupling Service, including the setup that is needed to run it as a service on the server. More advanced configuration changes can be done through a small number of configuration files, which are described in this section.

Folder Structure

The Coupling Service is installed in its own folder structure, much like, for example, the folder structure of a web server. By default MConfig puts this folder structure below the.

The following information describes the most important elements in this folder structure:

- `/configuration` — This folder contains Maconomy-specific configuration files:
  - `server.ini` — Configuration settings for the Coupling Service, such as the server name and port number for the Maconomy server and the port numbers for the Client service ports.
  - `maconomy.security.config` — Specification of JAAS login rules that are available to the Coupling Service.
  - `logback.xml` — Specification file for the built-in logging framework.
- `/log` — Default output directory for log files that are generated by the built-in logging framework.
- `/exportlog` — This folder contains a tool to export log files and system configuration as a zip/tar file.
- `/servicewrapper` — This folder contains the wrapper tool that is used for wrapping the coupling service as a Windows service. The configuration of the Windows service is generated by MConfig and stored in the file `servicewrapper/conf/wrapper.equinox.conf`.
- `/dropins` — This folder can be used for custom/third-party OSGi bundles. The contents of this folder are preserved during service pack installation.
- `/plugins` — This folder contains all of the OSGi bundles that are distributed as part of the CouplingService. It is normally not needed or recommended to make any manual changes to this folder, because its contents may be completely replaced during service pack installation.
- `/CouplingService(.exe)` — The platform executable for the Coupling Service.
- `/CouplingService.ini` — A file that contains command-line options for the Coupling Service.

- `/version.info` — A file that contains version information and compatibility rules for the Coupling Service. See the following information.

/configuration/server.ini

This file contains configuration settings for the Coupling Service. The format of the file is a number of lines, each of which contains one property value assignment, with the syntax "<property>=<value>."

The mandatory settings are:

- server.address=<name or IP address>, and server.port=<port-number> — the address and port number of the Maconomy Application server.
- web.port=<port-number>, and coupling.dirmi.port=<port-number> — These two ports are used by the Maconomy 2.0 Workspace Client to connect to the Coupling Service. Both of them must be accessible from all network locations where users are expected to connect. The web port specifies the Coupling Service's HTTP handshake port. This handshake can be viewed using a normal browser at the address http://<server.url>:<web.port>/handshake/1/

The most important optional settings are:

- server.max=<number> — The maximum number of server processes that the Coupling Service is allowed to start to serve requests. The default value is 10.
- server.pool=<number> — The number of server processes that the Coupling Service is allowed to keep in its server pool to serve future requests. The default value is 3.
- server.lifetime=<seconds> — The number of seconds that a server is permitted to exist before it is terminated. The default is 60.000 (about 16.6 hours).
- server.timeout=<seconds> — The number of seconds that a server will wait on a connection between commands. The default is 600 (10 minutes). The Coupling Service ensures that idle servers are kept alive, so this option only controls how long an active server should wait for a (non-responsive) client before disconnecting.
- coupling.dirmi.encryption=<true/false> — Controls whether the communication between the Coupling Service and clients is encrypted. The public and private keys should be specified using the following two settings: server.public-keystore and server.private-keystore.
- client.notifications.recalculation.interval=<seconds> — The interval between client requests for notification recalculation. The default is 1800 seconds, that is, 30 minutes.
- log.config=<path> — The path to the configuration file for the built-in logging framework.
- metrics.csv.enabled=<true/false> — Controls whether metrics should be written as comma-separated values (CSV) to a configurable file system location (specified by metrics.csv.location). The metrics feature is an extended form of logging that can be used for monitoring of the Coupling Service's service level.

A Technical Note about the Server Pool

The server pool is actually not just *one* pool, but a collection of pools, each of which serves a single database/language combination. In addition there is usually one extra server pool, for serving specific request types that should not occupy servers in the other pools.

Example

If a Coupling Service instance is serving two databases (a test system and a live system) with three languages installed, such as US, DK, and ES, there are 6 (2×3) server pools for serving each database/language combination plus the extra pool, for a total of 7 server pools.

The server pool parameters (server.pool and server.max) apply to every pool, so with the default settings, the system in the last example can have as many as 70 server processes running simultaneously, assuming that all database/language combinations are experiencing constant high load.

However, in the typical scenario a few of the server pools suffer most of the load, so the server pools should be dimensioned to handle the expected peak load of the most active server pools. When idle, the remaining pools scale down gracefully to hold at most server.pool idle servers, so this parameter should generally not be set too high.

/configuration/maconomy.security.config

This file contains the specification of JAAS (Java Authentication and Authorization Service) login rules that are available to the Coupling Service. The basic syntax of the file is:

```
<login-rule> {
    <module-path> <status-flag>
        <module-options>
}
```

The complete syntax is beyond the scope of this document, but it is possible to perform simple modifications to the file to enable various features, and these are described in the following sections.

Kerberos SSO (Single Sign On)

The following change is needed to enable Kerberos SSO for the Maconomy 2.0 Workspace Client. In the “Maconomy” login rule, there are two commented-out lines:

```
//org.eclipse.equinox.security.auth.module.ExtensionLoginModule
sufficient

//extensionId="com.maconomy.lib.coupling.MaconomySSOLoginModule";
```

To enable Kerberos SSO, the comment tag (//) should be removed from these lines.

The Maconomy server must, of course, be configured for Kerberos SSO as well, as described elsewhere.

Business Objects Login

The following change is needed to allow the Maconomy 2.0 Workspace Client to authenticate with the Business Objects server.

In the BusinessObjects login rule, the following parameters must be set to match the local system set-up: host, port, and authenticationType.

See the Business Objects documentation for additional details about the values of these parameters.

Note that currently you can only associate one Business Objects server with a Coupling Service installation. If the Coupling Service is connected to several databases, for example both a test and a live system, the same Business Objects server will be integrated into all of them. This limitation will be addressed in the future.

/configuration/logback.xml

This is the specification file for the built-in logging framework. See “Logging and Debugging” for details.

/CouplingService.ini

This file contains command-line options for the Coupling Service executable. It must be placed next to the executable and have the same name, except for the file extension. The syntax of the file is one argument per line.

This file is used for low-level arguments to the executable itself or the underlying Java Virtual Machine, such as the maximum allowed heap size and various other memory management characteristics.

This file is also used to set global system properties that can be used to control aspects of the platform that cannot be changed in any other way.

The set of properties that can be set in this way is huge, and requires in-depth technical understanding of the components that they relate to.

/version.info

A file that contains version information and compatibility rules for the Coupling Service. Compatibility rules state which specific client versions or version ranges are compatible with the Coupling Service.

The syntax is ALLOW/DENY <start> <end> where the ALLOW keyword is used to specify allowed versions, and the DENY keyword is used to specify denied versions. The <start> and the optional <end> tokens specify individual versions, for example, 15.0.3 for major version 15, minor version 0, and service pack 3. If no end version is specified the allow/deny rule is specific to an individual version.

Note that this file is overwritten during an upgrade.

Menu Dock

The Menu Dock—to the left in the Workspace Client—can be configured to show which system the WSC is connected to. This is most often used when having test systems, whereas most production systems do not show anything. The default is not to show anything.

The following shows an example of how the menu dock can be used.

The configuration of the content of the menu dock is done on the Coupling Service in the MCSL file. The configuration might look as follows:

```
<Binding namespace="client:menudock">
   <Fields>
      <Field name="Title" valueString="^{'Logged into: ' +
      shortname()}"/>
      <Field name="Color" valueString="rgb=#abba76" />
      <Field name="TopToBottom" valueString="false" />
      <Field name="FontHeight" valueString="8" />
   </Fields>
```

```
</Binding>
```

You can specify another color by assigning another field "Color" (spelled the US way) to a valueString that has one of the two formats:

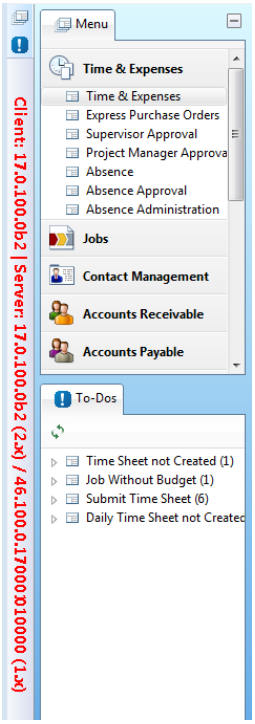
```
rgb=num,num,num (where num is a decimal number in the range 0-255)
```

OR

```
rgb=#xxyyzz (where xx, yy and zz are two-digit hexadecimal numbers in
the range 00-FF).
```

Other attributes that you can specify are:

- FontHeight, which indicates the font height to use. (The default is 12.)
- TopToBottom, which indicates whether the text is drawn top-to-bottom (this is the default) or bottom-to-top (if the value is specified as false).



Logging and Debugging

The Coupling Service has a built-in logging framework that can be used to log various events in the Coupling Service filtered on the source and severity level of each event. You can only log already declared events, meaning that you cannot retrieve information that the Coupling Service programmers have not already chosen to make available for logging.

Logging is controlled through a configuration file, by default located in configuration/logback.xml. The format of the file is XML, but it is relatively easy to edit using a normal text editor. The following information provides a brief introduction to the use of the logging framework. Additional information about the logback configuration file format can be found here: http://logback.qos.ch/manual/joran.html#syntax

The configuration file consists of two main sections: appenders and loggers.

An appender declares a destination for the log output. Typical output destinations include the console standard output and output files. Each appender specifies the desired format of the log entries, including parameter substitutions, as well as other settings, such as the rolling policy for splitting long log files on size or time interval.

A logger declares which events should be logged, as well as the appenders that they should be logged to. Loggers can be restricted on Java package path and the severity level of the event. The defined levels (from least to most verbose) are:

OFF, ERROR, WARN, INFO, DEBUG, TRACE, ALL

Example

During normal service an appropriate log level is typically to log everything at the ERROR level so that only error situations are logged. This is expressed in the following default logger rule:

```
<logger name="com.maconomy">
  <level value="ERROR" />
  <appender-ref ref="FILE" />
</logger>
```

If a reproducible error is observed in a module it is possible to escalate the level of logging for that module by adding an additional logger rule. This might yield useful information about the operations leading up to the error:

```
<logger name="com.maconomy.<path-to-module>" additivity="false">
  <level value="DEBUG" />
  <appender-ref ref="FILE" />
</logger>
```

The additivity parameter is used here to prevent the events from being matched by other loggers, which would otherwise result in multiple log entries for the event.

The default logback.xml file that is distributed with the Coupling Service contains several example loggers, with log level set to “OFF,” because examples of log events often prove useful when investigating runtime problems.

Export Log Tool

The log files, metrics, and configuration files of a given Coupling Service can be exported using the Export Log tool. This tool wraps all relevant log and configuration files as a zip/tar file that can be used to communicate system and configuration information during support cases. The tool is executed using either the .bat file or the .sh shell script, which are located in the `/exportlog/scripts` folder.

Monitoring and Tracing with JVisualVM

JVisualVM is a very useful tool that is distributed by Oracle as part of the standard Java Development Kit (JDK). With this tool you can connect to a running Java process on the current machine (and even on a remote machine) and extract all sorts of useful information, such as thread count, processor load, memory consumption, and much more.

In addition, you can install various plug-ins to greatly extend the capabilities of JVisualVM. Deltek recommends that all technical consultants familiarize themselves with this tool.

With a systematic top-down approach, the built-in logging framework and JVisualVM can be powerful tools to track down and understand otherwise unexplainable system behavior. However, sometimes the required log events are simply not available, usually because the original developer has not taken logging into consideration.

In these cases you might want to take a more “low-level” approach, and implement the Coupling Service directly using external tools. One such tool is the BTrace plug-in to JVisualVM. This plug-in is included in the standard JVisualVM distribution and can be installed with a few mouse clicks.

To use the BTrace plug-in you must first tell the Coupling Service boot-loader to allow BTrace access to the system internals. You do this by adding the following line to CouplingService.ini:

```
org.osgi.framework.bootdelegation=com.sun.btrace, com.sun.btrace.*
```

After you restart the Coupling Service and JVisualVM is connected, you can use BTrace to write custom probes that trigger on various events within the system, such as method calls, errors, and other events. These probes can be used to get detailed information about code execution paths that are not otherwise logged.

External Links

For more information about JVisualVM and the BTrace tool please consult their websites:

http://visualvm.java.net
http://kenai.com/projects/btrace.

Web Sockets (One Port)

Dirmi RPC over WebSockets (One Port)

The Workspace Client communicates with the Server 2.X through both a set of web services and the Dirmi RPC protocol.

The Dirmi RPC endpoint has been exposed on a separate TCP port, which made it necessary for customers to set up firewalls, proxies, and other infrastructure components to handle two distinct kinds of traffic: HTTP and Dirmi.

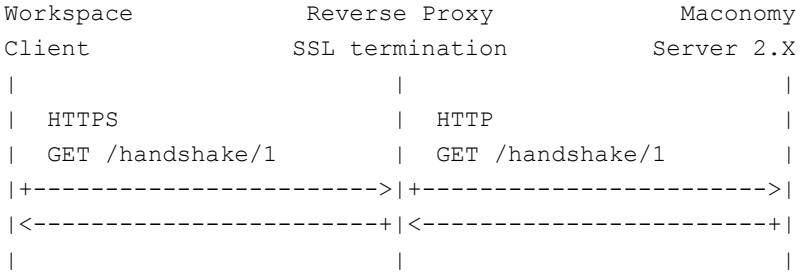
Maconomy 2.0 sp7, Maconomy 2.1.1 and later versions introduce support for the WebSocket protocol that allows the publication of the Dirmi RPC endpoint as a web service. This removes the need for two separate TCP ports, because everything is exposed as HTTP traffic. The WebSocket protocol is an emerging technology that enables full-duplex communication between clients and a web server.

When using WebSockets, encryption is delegated to an off-the-shelf SSL termination proxy, which can be software-based such as nginx or a hardware-based network appliance. This gives customers a standard, flexible way of administering their security setup.

For now, Dirmi RPC over the WebSocket protocol is not the default configuration for Maconomy, but it may become the default in future releases. The configuration of the WebSockets is handled by Mconfig; the following description is mainly for information, but may also be useful when special configurations are required.

Overview

The following is a diagram that shows the basic setup.



```
|  HTTPS                    |  HTTP                     |
|  GET /workspace-rpc       |  GET /workspace-rpc       |
|  Upgrade: websockets      |  Upgrade: websockets      |
|+------------------------->|+------------------------->|
|<-------------------------+|<-------------------------+|
|                           |                           |
```

- The workspace client communicates with the SSL termination proxy using HTTPS (encrypted)
- The SSL termination proxy forwards the requests using HTTP (unencrypted)
- Dirmi RPC happens using the WebSockets protocol, which will look like a long-running HTTPS request

Setup

To configure web sockets, the following setting in server.ini needs to be uncommented and changed to “websockets.”

```
# The transport protocol to use when exposing the Maconomy Coupling
# Service.
#
# Valid values are:
#
#   sockets     - Use plain sockets that are accepted on a separate
#                 port (as specified by the option coupling.dirmi.port)
#   websockets  - Use WebSockets that are accepted on a URL on the web
#                 service port
#
# The default is to use 'sockets'
#-------------------------------------------------------------------------
coupling.dirmi.transport=websockets
#-------------------------------------------------------------------------
```

The local encryption that happens in the Maconomy Server 2.X is NOT supported with web sockets, so local encryption MUST be disabled (see the following for how to configure encryption through an SSL termination proxy).

```
# Control whether to encrypt the communication between the Workspace
# Client and the Coupling Service.
#
# When encryption is enabled the following options are used to
# configure the encryption keys:
#   server.public-keystore
#   server.private-keystore
#
# The default is to encrypt.
#-------------------------------------------------------------------------
coupling.dirmi.encryption=false
#-------------------------------------------------------------------------
```

Setting Up an SSL Termination Proxy

To encrypt the traffic between the Workspace Client and the Maconomy Server 2.X, it is necessary to set up a reverse proxy that performs SSL termination. The client connects to the reverse proxy with an encrypted connection, and the proxy forwards the unencrypted connection to the coupling service.

The reverse proxy can be a software proxy such as nginx or Apache. It can be installed on the same machine as the Server 2.X or on a separate machine.

The reverse proxy can also be a dedicated network appliance.

Changes in server.ini

In this configuration the Maconomy Server 2.X must instruct the client to use encryption in the server handshake, but not perform the encryption itself.

The way to achieve this is by setting the following combination of options in the server.ini file:

```
coupling.dirmi.encryption=false
coupling.dirmi.proxy.encryption=true
```

See the Administration Guide section “Proxy with Server 2.X” for details.

Proxy Configuration: nginx

```
# HTTPS server for service
#
server {
   listen       <proxy-listen-port>;
   server_name  <server-name>;

   ssl                  on;
   ssl_certificate      /keystore/cscert.pem;
   ssl_certificate_key  /keystore/cskey.pem;

   ssl_session_timeout  5m;
   ssl_protocols  SSLv2 SSLv3 TLSv1;
   ssl_ciphers  ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
RSA-RC4-SHA:ECDHE-RSA-AES128-SHA:RC4-SHA:RC4-MD5:ECDHE-RSA-AES256-SHA:AES256-
SHA:ECDHE-RSA-DES-CBC3-SHA:DES-CBC3-SHA:AES128-SHA;
   ssl_prefer_server_ciphers   on;

   location / {
          proxy_pass http://<target-server>:<target-port>;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header Host $host;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

   }

   location /workspaces-rpc {
          proxy_pass http://<target-server>:<target-port>;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header Host $host;
```

```
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            # WebSocket support (nginx 1.4)
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_read_timeout 30m;
    }
}
```

Proxy Configuration: Apache 2.4

```
Listen <proxy-listen-port>
<VirtualHost *:<proxy-listen-port>>
    ServerName <server-name>

    <Proxy *>
       Order deny,allow
       Allow from all
    </Proxy>

    ProxyRequests  Off
    ProxyPreserveHost  On

    ProxyPass  /workspaces-rpc  ws://<target-server>:<target-port>/workspaces-
rpc  retry=0  disablereuse=On
    ProxyPass  /               http://<target-server>:<target-port>/
retry=0

    SSLEngine              On
    SSLProxyEngine         On
    SSLCertificateFile     /keystore/server.crt
    SSLCertificateKeyFile  /keystore/server.key
</VirtualHost>
```

Advanced Configuration

It is possible to override the default websocket endpoint URI in the server.ini:

```
# The URI where the Maconomy Coupling Service is exposed when using
# websockets transport.
#
# The path component of this URI will be used as the Servlet alias for
# the web sockets servlet.
#
# By default, the value of this configuration parameter is sent to
# clients by the handshake web service. To override the URI sent in the
# handshake, use the option: coupling.dirmi.client.websockets-uri
#-------------------------------------------------------------------------
# coupling.dirmi.websockets-uri=/workspaces-rpc
#-------------------------------------------------------------------------
```

By default the endpoint URI is sent to the client in the server handshake. However, if required, the URI sent to the client can be configured separately.

```
# The websockets URI as sent in the handshake web service.
#
# By default, the value of this configuration parameter is derived from
# the value of the option: coupling.dirmi.websockets-uri
#
# Use this option to override the URI sent in the handshake.
# This can be useful if for example an intermediate proxy is set up to
# perform URI rewriting.
#-----------------------------------------------------------------------
# coupling.dirmi.client.websockets-uri=/workspaces-rpc
#-----------------------------------------------------------------------
```

One example of a use case is to configure the reverse proxy to serve the handshake unencrypted (e.g. on port 8080) while the websockets endpoint is served encrypted (e.g. on port 443). This can be achieved with the following configuration:

```
coupling.dirmi.encryption=false
coupling.dirmi.proxy.encryption=true
coupling.dirmi.client.websockets-uri=wss://servername:443/workspaces-rpc
```

Note that wss:// is the secure websockets protocol that corresponds to https://, while ws:// corresponds to http//.

Background Task System Nature Setup

Setting up databases by type / nature allows you to control how background tasks are carried out in test and production databases. Since background tasks are copied as part of a database replication for testing purposes, if you do not take measures to control the background tasks, they will carry out as scheduled. This could lead to such things as inadvertently synchronizing data with third party systems, or sending customer emails from the test system as well as the production system. Instead, specify that a background task will only be picked up by databases of a certain nature, such as Production or Test.

To facilitate this feature, first set up the needed system natures in the configuration file, and then activate background task assignment to a system nature in Workspace Client.

Additionally, you can set up precautions for certain background tasks so that these items behave differently based on the nature to which they are assigned. Precautions exist for file location, URLs, and email accounts so, for example, that you do not inadvertently send a client an email from a test system, or sync integration data from a test system.

Setup System Nature

You must first set up database natures so that you can then assign them to background tasks.

Note: The system nature names are user-defined, though we recommend using names such **test**, **dev**, and **production**.

Additionally, all recommendations in this section define the *default* settings to reference the settings you want for the test nature. You must also define a specific setting for the production nature (and possibly other natures as well).

To set up database natures:

1. In the Coupling Service, go to the configuration file **server.ini**.
2. Enter a default nature to a system shortname that defines the configuration, such as **test** or **production**.

   For example:

   ```
   # default nature is test
   system.nature=test

   # the production database short name
   system.nature.macoprod=production


   # development databases belong to nature
   # development
   system.nature.dev1=development
   system.nature.dev2=development
   ```

   Note: While you may have more than one system shortname database of **test** or **dev** nature, you must have only one system assigned **production** nature. Additionally, assigning a system as test ensures that, unless a background task is assigned a different nature, it will default be assigned the **test** type.

Email Account Precaution

You can control email-related background tasks by associating them with systems of a certain nature, and the system nature assignment overrides the email behavior. However, as a precaution you can also set up default email accounts associated with test systems to ensure

that, if a setting is inadvertently missed, emails from a test system go to a pre-defined email account, such as an internal test account. This precaution ensures that, for example, you do not send test email to customers, as it overrides whatever email address is assigned to the background task.

To set up a default email account for a test system:

1. In the Coupling Service, go to the configuration file **server.ini**.
2. Find the email lines where you define email properties.
3. Enter a new email line to define that emails of a test nature go to a pre-determined email address, in the format:

```
email.to.nature.test=emailaddress@domain.com
```

To set up a specific mail server:

Set up a specific mail server for use with a specific nature. For example, set-up a mail-server that will redirect all mails to a test account.

1. In the Coupling Service, go to the configuration file **server.ini**.
2. Find the mail server lines where you define the properties.
3. Configure the setting in the following format:

```
mailserver.address=<ip-address-of-mail-server>:<port>

mailserver.address.nature.production=<ip-address-of-
production-mail-server>:<port>

# The mailserver address specifies the default mailserver
used

# by the batch runtime (may be overridden specifically for
batch)

# as well as the "EmailOnAction" action.

# This property can be specified globally, by system nature
and/or shortname.

# The global value can be overwritten by the system nature
value which again

# can be overwritten the the shortname value.

# If no value is specified, the system parameter
MailPortAndServer will be used

#

# The syntax is: serveraddress:portno

# e.g., 172.10.8.100:25 or mailserver.trifolium.com:25

# to use the mail server at IP-address: 12.10.8.100 at port
25

# or the mail server at mailserver.trifolium.com at port 25
respectively.

#-------------------------------------------------------------

# mailserver.address = <mailserveraddress>:<port>
```

```
#
# mailserver.address.nature.<system-nature-id> =
<mailserver.address>
#
# mailserver.address.<shortname> =
<mailserver.address.nature.<system-nature-id>>
#-------------------------------------------------------------
```

Third Party Integration Precaution

Set up a specific URL for test integrations so that an integration such as CRM does not inadvertently sync from a test database. This set up assumes that you have at least two integration instances, where one is used for testing and the other is used for production.

To set up an integration URL for a test system:

1. In the Coupling Service, go to the configuration file **server.ini**.
2. Find the lines where you define URL properties.
3. Enter a new line to define that an integration of a test nature syncs with a pre-defined URL set up for a test purpose.

   For example (for CRM integration):

   ```
   # by default use the test crm installation
   url.detekcrmwebservice=https://testcrm.trifolium.com/vision
   /visionws.asmx
   # for production, use this crm installation
   url.deltekcrmwebservice.nature.prodcution=https://livecrm.trifoli
   um.com/vision/visionws.asmx
   ```

   Note: The specific integration must have been prepared for reading specific url-names.

File Location Precaution

Set up file locations based on system nature if you have a set up where an external process generates file to import or where output documents produced by background tasks need to be stored in different locations depending on the system nature.

To set up a default file location for a test system:

1. In the Coupling Service, go to the configuration file **server.ini**.
2. Find the lines where you define URL properties.
3. Enter a new line to define that an integration of a test nature syncs with a pre-defined URL set up for a test purpose.

   For example:

   # the default location for “postings” file reference
   filepath.postings=c:\maconomyoutput\test\postings

   # for production, the location should be different
   filepath.postings.nature.production=c:\maconomyoutput\postings

Activate Background Task Rules

After you set up the system natures in the configuration files, you can assign individual background tasks as well as individual schedule rules to a system nature in the Workspace Client. **Background Tasks > Setup > List of Schedule Rules.**

Advanced Logging

This document describes the use of import programs for Maconomy 2.0+. Before you proceed, make sure that your import program and program-specific import manual match the same Maconomy version and service pack number as this document, as a mismatch may cause errors in your import. This document describes the central concepts involved in the import process and explains the possibilities offered by import programs. Furthermore, this manual sets up a number of examples for the practical use of import programs. The concept definitions and examples are described on a general level. See *Getting Started* for an overview of working with a specific import program.

Please note that the functionality described in this manual only applies to those Maconomy import programs that are generated according to the Standard Import Format. Maconomy's product range also includes a number of import programs with functionality that is not covered in this manual. You should therefore check the manual for the import program in question to see whether the program was generated using Maconomy's Standard Import Format.

Introduction

Maconomy contains a suite of data security features. One feature is the logging system, which is designed to comply with various legal requirements.

From a security point of view, two areas are especially interesting to monitor:

1. Which users change data in the system, what do they change, and from what?
2. What data do which users view, print, export, and search for in the system? There may be legal requirements to monitor, such as who changes the credit maximum of a customer or who extends a credit grace period.

Maconomy has a powerful method of keeping track of any updates and queries in the system. This is described in the following sections.

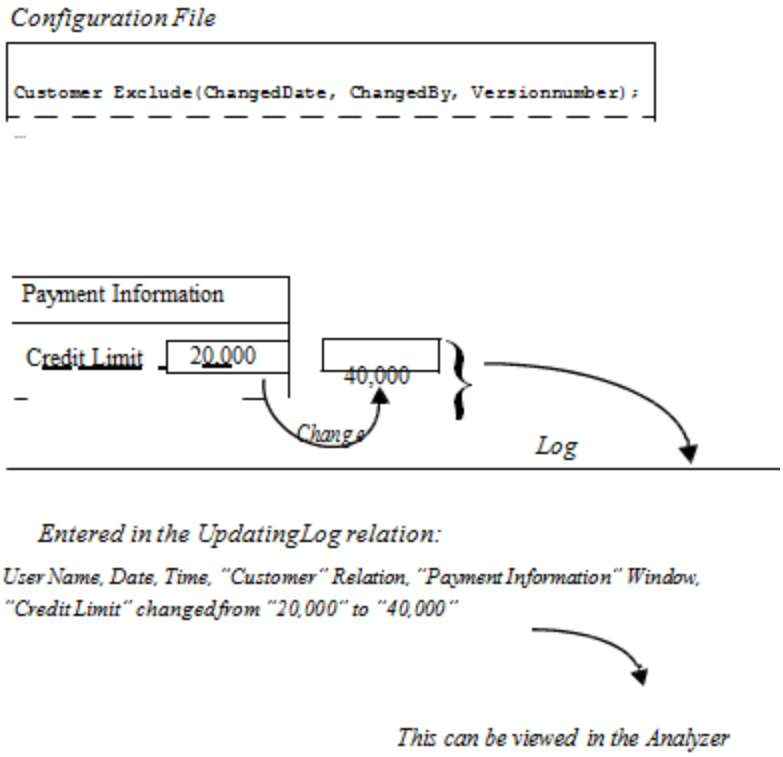
General Functionality

The kind of information to be logged is specified in a configuration file in which the system administrator can define the exact relations and fields on which logging should be performed. To find the names of relations and fields in the database, please consult Database Description in the Reference Manual for the current version of Maconomy.

By logging only the data changes that require monitoring, storage space can be limited, and response times optimized. After the relevant logging rules have been set up, all of the monitored changes and queries are stored in relations in the Maconomy database.

The log information is viewed by means of the Maconomy Analyzer, using either the Update Log Analyzer Report or the Query Log Analyzer Report.

The following diagram shows the flow of information through Maconomy when a user changes a relation that is monitored by the Update Log functionality:



Update Log

The Maconomy Update Log is used for registering changes made in various relations in the Maconomy database. Maconomy does not log the entire relation when something is changed. Only those fields that are actually changed are logged.

An entry is made in the log for a selected relation when a user makes a direct change to the relation. However, transaction updates performed by Maconomy are not logged. For instance, if all fields in the Customer relation are monitored, Maconomy records an entry in the log if a user changes the customer's credit maximum in the Payment Information window. But if a sales order is approved and billed, and the customer's balance is thus increased, no log entry is made.

When deleting an entry, Maconomy can log either the fact that the entry was deleted, or the value that was deleted. The method of logging deletions depends on the keyword LogOnDelete.

Setup

Configuration File

The relations and fields to be monitored are specified in a text file on the server. This file is encrypted to prevent unauthorized changes (tampering) by users. The encryption is performed by Maconomy A/S on request.

- **Location** — The configuration file is called UpdateLog.cnf and is placed in the folder Definitions on the Maconomy server.
- **Content** — Two optional keywords can be added to the beginning of the file: NoLogOnCreate and LogOnDelete. The order of the keywords is insignificant.
  - **NoLogOnCreate** — If this keyword is specified, log entries are not made when a record of the types specified later in the file is created, only when it is changed.

- **LogOnDelete** — If this keyword is specified, Maconomy logs the deletion of a field in one of the included relations by registering the value that was deleted as a change to a blank value. If not specified, Maconomy simply records the fact that it was deleted (but does not include the deleted value).

In addition, enter the names of the relations that you want to log when they are updated. Each line must end with a semicolon. You can modify the relation selection in several ways:

- You can add lists of fields that you want to exclude from the log, for example, ChangedBy, which is included in the log anyway. This is done using the keyword Exclude.
- If you only want to log changes to a few fields in a relation, you can add lists of fields that you specifically want to add to the log. This is done using the keyword LogOnly.
- You can specify a custom key for the relation. It is not always interesting to use the database specified key. For instance, in the relation DialogGroupMember, the database key is GroupTitle, LineNumber, but it would be more interesting in the subsequent use of the Analyzer report to be able to use GroupTitle, WindowTitle as key. This is done using the keyword Key.
- You can specify that you only want to log an entry if one or more specific fields are changed. For instance, you may only want to log changes to a vendor if the vendor's bank account number is changed. This is done using the keyword IfChanged.

BNF Syntax

The formal syntax for the specification of relations and fields can be expressed as follows in Backus Naur Form (BNF):

```
<UpdateLog> ::= [NoLogOnCreate;][LogOnDelete;]<RelationDefinition>
    {; <RelationDefinition>}
<RelationDefinition> ::= <RelationName>
       (                    Key
       <FieldList>) ( IfChanged
       <FieldList>) ( Exclude
       <FieldList>)
       (                    LogOnly
       <FieldList>)
<FieldList> ::=      (<FieldName> {; <FieldName>})
```

Example

The following configuration file example sets up Maconomy to log changes in the following way:

- No log entries are added when an object is created, only when changed.
- When a logged field is deleted, the deleted value is logged as a change to <blank>.
- In the Customer Information Card window (and related windows), changes to the Credit Max., Segment, and Customer Payment Mode fields are logged.
- In the Vendor Information Card (and related windows), System Information, Company Information, Users, Access Information, and Access Levels windows, all changes are logged with the exception of the Changed On, Changed By, and Version No. fields.
- In addition, all changes to the following windows are logged: Groups (with a custom key), Access Level Lines (table part of Access Levels), and User Access Levels.

The file would look like the following example:

```
NoLogOnCreate;
LogOnDelete;
Customer            LogOnly(CreditMax,
                    Segment,
Creditor            Exclude(ChangedDate, ChangedBy, Versionnumber);
SystemInformation   Exclude(ChangedDate, ChangedBy, Versionnumber);
CompanyInformation  Exclude(ChangedDate, ChangedBy, Versionnumber);
User                Exclude(ChangedDate, ChangedBy, Versionnumber);
DialogGroupMember   Key(GroupTitle, WindowTitle);
AccessInformation   Exclude(ChangedDate, ChangedBy, Versionnumber);
UserDialogGroup;
AccessLevelHeader   Exclude(ChangedDate, ChangedBy, Versionnumber);
AccessLevelLine;
UserAccessLevel;
```

Notes

What Is Logged

- **Data** — A log entry is stored in two relations in the Maconomy database: UpdatingLog and UpdatingLogLine. The former contains the name of the relation and the window in which the change was made and the key of the item that was changed. The latter contains information about which fields in the relation were changed.

  The following data is logged in the UpdatingLog relation:

  - The name of the user who changed data
  - An update log number
  - The date of the change
  - The time of the change
  - The name of the relation that was changed
  - The name of the window in which the relation was changed
  - The key (possibly composite key) of the changed data

  The following data is logged in the UpdatingLogLine relation:

  - The name of the user who changed data
  - An update log number
  - An update log entry number
  - The name of the field that was changed
  - The content of the field before the change
  - The content of the field after the change (possibly blank, if LogOnDelete is specified)

The first three items in the preceding list constitute the key reference to the UpdatingLog relation.

A log entry might look like the following at the record level:

UpdatingLog

NameOfUser	Updating LogNumber	TheDate	TheTime	RelationName	WindowName	KeyField Name1	KeyValue1	...
Arod Hasufel	1	040219	10:23	Customer	PaymentInformation	Customer Number	31003100	

And at the field level:

UpdatingLogLine

NameOfUser	UpdatingLogNumber	LineNumber	FieldName	CorrectedFrom	CorrectedTo
Arod Hasufel	1	1	CreditLimit	20,000	40,000

The Analyzer report Update Log is built on these two relations.

Query Log

The Maconomy Query Log is used for registering which users query the Maconomy database, and what information they see, print, or export.

The Maconomy database is considered to be queried in the following cases:

- When viewing information in a dialog
- When using the Find window
- When printing using parameter dialogs
- When printing using Print This in the File menu
- When printing using an action in the Action menu
- When exporting data from dialogs, including Find windows
- When exporting data using an action, for example, Create Payment File
- When exporting data using add-on programs
- When running SQL (RGL) reports

However, internal transactional lookups are not considered a query. For instance, when Maconomy checks for the existence of a customer in the General Journal, no log entry is made.

Setup

Configuration File

The relations and fields to be monitored are specified in a text file on the server. The file is encrypted to prevent unauthorized changes (tampering) by users. The encryption is performed by Maconomy A/S on request.

- **Location** — The configuration file is called `RequestLog.cnf` and is placed in the folder `Definitions` on the Maconomy server.

- **Content** — In the file, enter the names of the relations that you want to log when they are queried. Each line must end with a semicolon. You can modify the relation selection to add a specific key to the log.
- **Input File Encoding** — This field is in the parameter dialogs for auto-generated import programs. The encoding can be specified either in this field or on the first line in the import file. Valid encodings can be found here:

  http://demo.icu-project.org/icu-bin/convexp

  Specifying in the import file must be done with a comment line (a leading tab) such as:

  ```
  DefaultInputCharSet=iso-8859-5
  ```

  Only some encodings, such as ansi, iso-8859-15 and utf-8, can be specified in the file. UCS-2/UTF-16 cannot be specified in the file. All supported encodings can be specified in the parameter dialog field.
- **BNF syntax** — The formal syntax for the specification of relations and fields can be expressed as follows (in Backus Naur Form):

  ```
  <QueryLog> ::= <RelationDefinition>{; <RelationDefinition>}
  <RelationDefinition> ::= <RelationName> ( Key<FieldList>)
  <FieldList> ::=      (<FieldName> {; <FieldName>})
  ```
- **Example** — The following example sets up Maconomy to log all queries in the Customer Information Card (and related windows), Vendor Information Card (and related windows), and the Job Budgets (and related windows) windows.

  ```
  Customer;
  Creditor;
  JobBudget;
  ```

Notes

What Is Logged

- **Data** — A log entry is stored in the Maconomy database relation QueryLog. This relation contains the name of the relation, the window in which the query was made, and the key of the item that was queried.

  The following data is logged in the QueryLog relation:

  - The name of the user who queried the data
  - The date of the query
  - The time of the query
  - The name of the relation that was queried
  - The name of the window from which the relation was queried
  - The key (possibly composite key) of the queried data

  A log entry might look like the following:

QueryLog

NameOfUser	TheDate	TheTime	RelationName	WindowName	KeyFieldName1	KeyValue1	...
Arod Hasufel	040219	10:23	Customer	PaymentInformation	CustomerNumber	31003100	

The Analyzer report Query Log is built on the QueryLog relation.

Query Types

Logging in Various Circumstances

The following describes what triggers a log entry in the database in various situations in Maconomy. This description assumes that logging is set up for the relations that are described in the examples.

- **Dialogs** — In a dialog, for example, the Customer Information Card, a log entry is made every time that an information card is shown to the user. This means when a user opens the window, browses to another information card, or selects a customer from a Find window.

  Redrawing the window is not regarded as a new query.

  If a user has opened two windows to the same relation, for example, the Customer Information Card and the Payment Information window, and the windows are linked (Automatic Find is turned on in the Maconomy client), two entries are made every time that the user browses any of the windows.
- **Find Windows** — The key of all entries that are shown in the Find window is logged. The log only registers the records that are actually shown. If, for instance, the user only sees the first 30 records in the Find window, only 30 records are logged, even though the search criteria actually resulted in hundreds of records.
- **Printing from Parameter Dialog** — When printing, all selections that are to be printed (based on printable items in the current print layout) are logged.
- **Printing Using Print This** — This works as when printing from a parameter dialog. In addition, the record from which Print This was selected is also recorded. This record is transferred from the dialog (not retrieved from the database).
- **Printing Using Actions** — Printouts that are invoked from the Action menu, such as Print Invoice in the Sales Orders window, are run (and therefore logged) in the same way as printouts that are selected using Print This.
- **Exporting Data from Dialogs or Find Windows** — In all dialogs where a relation is associated with the card part, you can export records from the associated relation (either by selecting Export… from the File menu or selecting Export Result in the Find pane). The exported records are logged in the same way as if they had been displayed in a Find window or in a dialog.
- **Exporting Data Using Actions** — In a number of Maconomy dialogs, you can export data using an action from the Action menu, for example, Create Payment File in the Payment Files window. In this kind of query, the user enters selection criteria while the export file is open, and some of the exported data is filtered to the file. It is impossible for the query log to tell which data is actually exported to the file, as this is part of the logic of the particular Maconomy window, so all of the data from the associated relations is registered in the query log.
- **Exporting Data Using Add-On Programs** — This corresponds to selecting an action in the Action menu.
- **Running SQL Reports** — This corresponds to exporting data using add-on programs.

You can use bind variables whenever it makes sense, such as for filter queries and notification queries.

Silent Sign-In Overview

This section describes Silent Sign In, which uses Maconomy as an external verifier for Business Objects. When Silent Sign In is configured, report request tickets are created and stored in the Maconomy system and passed to a Business Objects Silent Sign In component. The Silent Sign In component subsequently validates the ticket against Maconomy, and on successful validation executes the report.

When Silent Sign In is configured on the Maconomy and Business Objects (BO) systems, it allows Maconomy users to let their Maconomy password be used as the access to the Business Objects system.

Maconomy and Business Objects have their own, separate systems with user accounts and authentication. When a user logs in to Maconomy and requests a Business Objects report, the user must log in to the BO system. With Silent Sign In configured, the Business Objects login page is not displayed, and the report opens directly.

The instructions in this manual can be used for embedding BPM report windows **in the Maconomy Portal only**.

You can also embed BPM portal windows in the 2.x Workspace Client, but that uses a different technology (although some of the steps on the BO server are overlapping) and is described in “Trusted Authentication with the Workspace Client.”

Silent Sign In is based on a ticket-per-request method, where a Maconomy Portal MScript component creates and stores a unique report request ticket in Maconomy. The ticket and Maconomy user ID are added as parameters to the BO Infoview OpenDocument link, which loads and executes a Java servlet component on the BO server.

The servlet component validates the ticket by calling a Maconomy web service, which looks up the ticket in Maconomy for the specified user. If the ticket exists in Maconomy, and has not expired, a BO user login ID is returned to the BO servlet component. The Maconomy web service API removes the ticket on validation, preventing the ticket from being reused (or abused) by other users.

On successful validation of the ticket, the servlet component creates a BO session for the BO user, using BO’s Trusted Authentication method. Trusted Authentication is a standard feature of BO, where a hidden/secret password is assigned to the server and the client that is running Silent Sign In servlet component, which allows the servlet component to log in and create a BO session for a specific user in the BO system.

When the servlet has successfully logged in and created the BO session for the BO user, the report is executed. If ticket validation fails, or if creation of the BO session fails, the servlet component redirects the user to the standard BO Infoview login page.

Requirements

The following requirements exist:

- Maconomy Portal Framework
- Maconomy Web Services Framework
- Business Objects BI Platform 4.1

Bypassing Silent Sign In

The configuration and setup of Silent Sign In is an extension to the existing Maconomy and Business Objects (BO) systems, and only applies for report requests directed to the Silent Sign In servlet component on the BO server.

Report requests based on the Silent Sign In method use a different OpenDocument link compared to the standard link:

```
http://.../BOE/OpenDocument/opendoc/openDocumentSSI.jsp?...
```

The openDocumentSSI.jsp (notice the ***SSI)*** directs the request to the Silent Sign In component. The standard openDocument.jsp link still directs to the standard BO login page.

System Overview

The Business Objects (BO) Silent Sign In authentication method is based on a BO servlet Java component, which is used to authenticate a given report request. The Silent Sign In servlet component for Maconomy authentication is installed on the BO server, and the Tomcat web server is configured to execute the servlet component on a given report request.

Business Objects Servlet Component

The servlet component receives a ticket and user ID as parameters to the report request, and then calls a Maconomy web server API for validation of the ticket. When the ticket is validated, a BO user ID is returned from the Maconomy web service API call. The servlet component uses the user ID to log in to the BO system and create a BO session for the report request.

Maconomy Ticket

The ticket for the report request is created and stored in Maconomy when a Portal MScript Component is executed by a user request. The MScript component uses an MScript package named SSIsessionHandler for the ticket creation, where the function createTicket() creates unique tickets in Maconomy. Tickets are stored in Maconomy using MScript Common Values, where the name of the Common Value is in the format “SSIsessionTickets.*Username*”. *Username* is the login name of the Portal user.

When a ticket is created and stored in Maconomy, time stamp information about the creation is also saved. The ticket is only active and valid from the time when the ticket is created by the MScript component until the BO Silent Sign In servlet component validates the ticket by calling the Maconomy web service API. The validation removes the ticket from Maconomy, which prevents other users from reusing (or abusing) the ticket. The maximum lifetime of a ticket has a default of 5 seconds, which is the maximum time allowed from the creation of the ticket until it is validated. The default maximum lifetime can be changed if required.

Maconomy and Business Objects User ID Mapping

When the Silent Sign In servlet component calls the Maconomy web service API for ticket validation, a BO user ID is returned from the web service API on successful validation. The user ID that is returned is used by the servlet component to log in to the BO system and create a BO session for the report request.

The BO user ID is either the same as the Maconomy user ID or is set to a fixed user ID. A fixed user ID is optionally assigned to the ticket on creation, when the Portal Component calls the createTicket() function and sets the optional parameter BOuserName. If the ticket is created without a user ID assigned, the user ID that is returned is set to the same as the Maconomy user ID.

Access to Maconomy and Business Objects

Required Access to the Business Objects System

The creation of the BO session that is handled by the Silent Sign In servlet component requires Trusted Authentication to be enabled on the BO system. The setup of Trusted Authentication includes a Shared Secret password, which is a password that the Silent Sign In servlet component should use when requesting that session be created.

The Shared Secret password for servlet component is configured in the Central Management Repository (CMS) and downloaded as a file named TrustedPrincipal.conf on the BO server.

Required Access to the Maconomy System

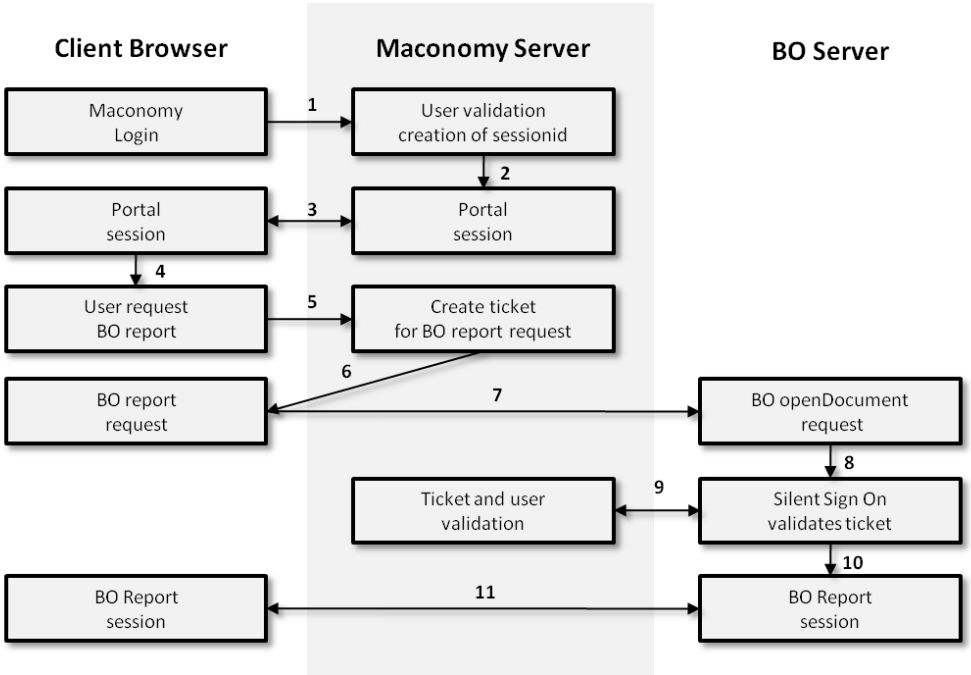
The creation of the ticket in Maconomy is done by the Portal MScript Component, which has the required access to the Maconomy system through current active Portal session.

The validation of the ticket is requested by the Single Sign In servlet component, which needs a Maconomy username and password to access Maconomy. The Maconomy user can be any user in Maconomy, and only needs access to log in to the system. No specific access to dialogs or access levels is required.

The username and password for the Maconomy web services and the Maconomy system is configured in the web.xml configuration file on the BO server.

Silent Sign In Request Flow Diagram

The following request flow diagram illustrates the Silent Sign In method; from the Maconomy Portal user login through a Business Objects (BO) report request and validation of the ticket.



The request flow can be explained as follows:

1. The username and password are passed from the browser client to the Maconomy Portal.
2. The user is validated against Maconomy, and a Portal session ID is created.

3. The user can now access the Portal using the session ID.
4. The user clicks the BO report link, which is a link to a Portal MScript Component.
5. The MScript component request is sent to the Maconomy Portal, which creates a unique ticket that is stored in Maconomy.
6. The current Portal username and ticket are added as parameters to the BO report link and passed to the client. This request is a java script redirection to the BO server, which is hidden from the user.
7. The BO request link openDocument1.jsp is sent to the BO server with parameters macusr=xxx and macticket=yyy.
8. The BO server starts the Silent Sign In servlet component, which reads the user ID and ticket from the link parameters.
9. The servlet component calls to the Maconomy web service function validateTicket.
10. The Maconomy web service validates the user, removes the ticket, and returns the BO user ID on successful validation. An error code and message are returned on failure.
11. The servlet component connects to the BO system using Trusted Authentication, logs in to BO as the BO user ID, and creates the BO session on successful validation. On validation failure, the component redirects to the default login page.
12. BO session is established between the client and the BO server.

Setup

This section describes how to set up Maconomy and Business Objects to use Silent Sign In authentication. The setup procedure includes configuration changes and file installations on the Maconomy Portal, Maconomy Web Service Framework, and the Business Objects server, including guidelines about how to enable Trusted Authentication. In addition, the setup information includes an MScript example of how to implement report requests in a Maconomy Portal MScript Component.

Installation Files

The following installation files are required for the setup of the Silent Sign In feature.

Folder and Files	Description
com\inovista - OpendocFilter.class - OpendocServletRequest.class - ValidateUser.class - ValidateUser$1class - OpendocFilter$ValidationInfo.class - ValidateUser$ReturnPackageParsingResult.class - ValidationLogger.class	Business Objects Silent Sign In servlet Java component
com\maconomy - SilentSignIn.class - SilentSignInLocator.class - SilentSignInPortType.class - SilentSignInSOAPBindingStub.class - ValidateTicketForBusinessObjects.class - ValidateTicketForBusinessObjectsResponseType.class - ValidateTicketForPeoplePlanner.class - ValidateTicketForPeoplePlannerResponseType.class	Business Objects servlet Java components for Web Service client

Folder and Files	Description
opendoc - openDocument1.jsp - web.template.xml - SSI-JarFile_List.txt	Business Objects Silent Sign In openDocument link and template web.xml file

Trusted Authentication

Trusted Authentication must be enabled on the Business Objects (BO) server, which is documented in the Business Objects Enterprise Administrator's Guide » Enabling Trusted Authentication.

The configuration steps include setting up the server and client, which in the case of the Silent Sign In servlet component is on the same server (the BO server).

The following steps are an example of how to configure Trusted Authentication, which is based on the BO Enterprise Administrator's Guide found here:

http://help.sap.com/businessobject/product_guides/sbo41/en/sbo41sp7_bip_admin_en.pdf

To configure Trusted Authentication, complete the following steps:

1. On the BO server, log in to the Central Management Console (CMC) with administrative rights.
2. Go to the **Authentication management area** of the CMC Client on the Enterprise tab and scroll down to the Trusted Authentication section.
3. Under **Trusted Authentication**:
    a. Click **Trusted Authentication is enabled**.
    b. Click **New Shared Secret**.
    c. Click **Download Shared Secret**. The File Download dialog box appears.
    d. Click **Save** and save the TrustedPrincipal.conf file to:

    ```
    <INSTALLDIR>\SAP BusinessObjects Enterprise XI 4.0\win32_x86\
    ```

    There is a chance the generated Shared Secret is too long that it causes an unnecessary line break. If this happens, generate a new Shared Secret.

    e. Enter the number of days for the Shared Secret Validation Period and timeout for the Trusted Authentication request. The recommendation is 1 day and 10000 milliseconds, but refer to the BO manual for details.
4. Click **Update** to save the settings.
5. Create a global.properties file in
    - `<INSTALLDIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\warfiles\webapps\BOE\WEB-INF\config\custom`

    with the following parameters:

    ```
    sso.enabled=true
    trusted.auth.user.retrieval=REMOTE_USER
    trusted.auth.shared.secret=
    ```
6. Redeploy BOE.war:

a. Open a command prompt

b. Change directory to:

```
<INSTALLDIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\wdeploy
```

c. Run **wdeploy.bat** with the following parameters:

```
tomcat7

-Das_instance=<server_name>

-Das_service_name=<Tomcat_service_name>

-Das_dir="<INSTALLDIR>\SAP BusinessObjects\tomcat"

-Dwork_dir="<INSTALLDIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\wdeploy\workdir"

-Dapp_source_tree="<INSTALLDIR>\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\wdeploy\workdir\tomcat7\application"

-DAPP=BOE

deployonly
```

7. Stop tomcat web server
8. Delete the "work" folder in
   - `<INSTALLDIR>\SAP BusinessObjects\tomcat`
9. Start the Tomcat web server.

Trusted Authentication with Workspace Client

This section describes how to set up Trusted Authentication for use by the Coupling Service and Workspace Client.

The configuration steps include setting up the server and client, which in the case of the Silent Sign In servlet component are on the same server (the BO server).

The following steps provide an example of how to configure Trusted Authentication, which is based on the BO Enterprise Administrator's Guide found here:

http://help.sap.com/businessobject/product_guides/sbo41/en/sbo41sp7_bip_admin_en.pdf

To set up Trusted Authentication for use by the Coupling Service and Workspace Client, complete the following steps:

1. On the BO server, log in to the Central Management Console (CMC) with administrative rights.
2. Go to the Authentication management area of the CMC.
3. Click on the **Enterprise** tab and scroll down to the Trusted Authentication section.
4. Under **Trusted Authentication**:
   a. Click **Trusted Authentication is enabled**.
   b. Click **New Shared Secret**.
   c. Click **Download Shared Secret**. The File Download dialog box appears.
   d. Click **Save** and save the TrustedPrincipal.conf file to:

```
<INSTALLDIR>\SAP BusinessObjects Enterprise XI 4.0\win32_x86\
```

There is a chance the generated Shared Secret is too long that it causes an unnecessary line break. If this happens, generate a new Shared Secret.

e. Enter the number of days for the Shared Secret Validation Period and timeout for the Trusted Authentication request. The recommendation is 1 day and 10000 milliseconds, but refer to the BO manual for details.

5. Click **Update** to save the settings.

After this is done you must enable Trusted Authentication on the Coupling Service. This is done by updating the OSGi settings in MConfig. A checkmark here enables Trusted Authentication, and then you must enter the Password from the preceding list.

Business Objects Setup

The following steps are required on the Business Objects (BO) server for installation of the Silent Sign In component.

To install the Silent Sign in component on the BO server, complete the following steps:

1. Copy the following folders:

   ```
   com\inovista
   com\maconomy
   ```

   to the Business Objects programs folder:

   ```
   <INSTALLDIR>\SAP BusinessObjects\tomcat\webapps\BOE\WEB-INF\classes
   ```

2. Copy the Shared Secret file **TrustedPrincipal.conf** from:

   ```
   <INSTALLDIR>\SAP BusinessObjects Enterprise XI 4.0\win32_x86
   ```

   To the Business Objects programs folder:

   ```
   <INSTALLDIR>\SAP BusinessObjects\tomcat\webapps\BOE\WEB-
   INF\classes\com\inovista
   ```

3. Copy the jar files from folder:

   ```
   jar files\
   ```

   to the Business Objects programs folder:

   ```
   <INSTALLDIR>\SAP BusinessObjects\tomcat\webapps\BOE\WEB-INF\lib
   ```

4. Copy the file:

   ```
   opendoc\openDocumentSSI.jsp
   ```

   To the Business Objects Programs folder:

   ```
   <INSTALLDIR>\SAP BusinessObjects\tomcat\webapps\BOE\WEB-
   INF\eclipse\plugins\webpath.OpenDocument\web\opendoc
   ```

5. Make a backup of the following file in the Business Objects Programs folder:

   ```
   <INSTALLDIR>\SAP BusinessObjects\tomcat\webapps\BOE\WEB-INF\web.xml
   ```

   A later restore of the web.xml file will disable the Silent Sign In component.

6. Copy the context of the following file:

   ```
   opendoc\web.template.xml
   ```

and insert the context into the following file in the Business Objects Programs folder, after the last <servlet-mapping > … </servlet-mapping > section:

```
<INSTALLDIR>\SAP BusinessObjects\tomcat\webapps\BOE\WEB-INF\web.xml
```

7. In the inserted lines, replace the following settings:
   - **WS-ID** – Maconomy Web Service ID
   - **CMS-SERVER** — Hostname of the BO CMS server.
   - **MACONOMY-WS** — Link to the Maconomy Web Service Framework (which can be found in the Maconomy\index.html that is generated by MConfig).
   - **MACONOMY-USER** — Login username for the Maconomy system.
   - **MACONOMY-PASSWORD** — Login password for the Maconomy system.
   - **SHARED-SECRET** – File name of the shared secret download from BO server (TrustedPrincipal.conf by default)

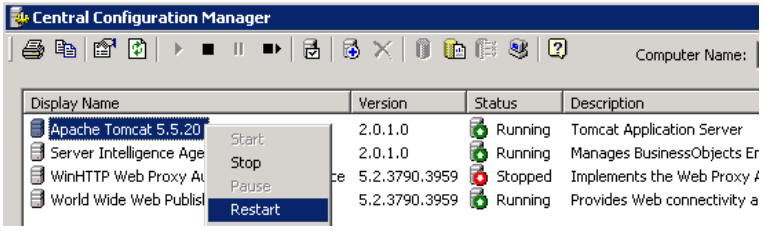
   The user in the Maconomy system does not need to have Administrator rights.
8. Save the changes to the web.xml file, and verify that the <filter> definition of the Silent Sign In component is similar to the following:

```
<!-- Silent Sign In with Maconomy external verifier -->
<filter>
     <filter-name>OpendocTokenManager</filter-name>
     <filter-class>com.inovista.OpendocFilter</filter-class>
     <init-param>
        <param-name>LogLevel</param-name>
        <param-value>1</param-value>
     </init-param>
     <init-param>
        <param-name>MacWS</param-name>
        <param-value>WS-ID</param-value>
     </init-param>
     <init-param>
        <param-name>validateURL</param-name>
        <param-value>MACONOMY-WS/soap.ms?service=SilentSignIn</param-
value>
     </init-param>
     <init-param>
        <param-name>adminUser</param-name>
        <param-value>MACONOMY-USER</param-value>
     </init-param>
     <init-param>
        <param-name>adminPassword</param-name>
        <param-value>MACONOMY-PASSWORD</param-value>
     </init-param>
     <init-param>
        <param-name>cmsRepo</param-name>
        <param-value>CMS-SERVER</param-value>
     </init-param>
     <init-param>
```

```
            <param-name>sharedSecretFilename</param-name>
            <param-value>SHARED-SECRET</param-value>
         </init-param>
      </filter>

      <filter-mapping>
         <filter-name>OpendocTokenManager</filter-name>
         <url-pattern>/OpenDocument/opendoc/openDocumentSSI.jsp</url-pattern>
   </filter-mapping>
```

9. Click **Start » Programs » SAP Business Intelligence » SAP BusinessObjects BI Platform 4 » Central Configuration Manager** to open the Central Configuration Manager on the BO server and restart the Tomcat Application Server:



Testing and Debugging

For testing and debugging purposes, the following are recommended to identify and trace the ticket and BO report request flow.

To use recommended settings for testing and debugging, complete the following steps:

1. Check the following file in the Business Objects Programs folder on the Business Objects server:

   ```
   <INSTALLDIR>\SAP BusinessObjects\tomcat\logs\stdout.log
   ```

   The Silent Sign In servlet component writes request information to this file and reports successful requests as well as failures. You can increase the log level that the Silent Sign In component produces in the web.xml file that was mentioned previously in this document.

2. Add debugging messages to the Portal MScript Component, which runs Java script alerts to show the exact link to the BO report request, prints information to standard output, or to log files on the Maconomy web server.

3. Add the following configuration setting to the MaconomyPortal.xxx.I in the cgi-bin/Maconomy folder on the Maconomy web server:

   ```
   log = all;tab
   ```

   All incoming requests will be logged to the MaconomyPortal.xxx.log file.

Security Overview

This chapter describes the security model that is implemented in Maconomy and some tips to increase the security of your Maconomy system.

Maconomy's security model, or access control system, is active in all modules in Maconomy, and is based on the user who is logged in. There are two types of access control in Maconomy:

- One for defining each user's access to opening windows and changing data
- One for defining what records in the database each user should be allowed to access

Access Control Elements

When designing the access control for your Maconomy system, you should be aware of all the elements of the Maconomy security model. Consider the following:

- Access groups — Each user is a member of at least one access group. A group is a specification of which Maconomy windows a member of the group is allowed to open, and whether the member is permitted to read, create, update, or delete information in the window in question.

  Note that a user's membership of the access group also defines the user's access to database views (defined for each access group in the View Groups window) and window layouts (defined for each access group in the Window Layout Groups window).

- Window layouts — By assigning different window layouts to different user groups (as mentioned previously), you can restrict access to certain fields and features in Maconomy.

- Actions — For each user, you can specify access to various actions in the Actions window. For instance, you can grant or deny permission to approve sales orders, print invoices, and so forth.

- Access levels — Access levels are used for managing user access to database entries.

  While groups determine whether a user can open a given window and change information, access levels determine the records that are available to the user in each window.

  Data can be organized in an access hierarchy, in which the access level assigned to each user determines whether he or she has access to the data. For instance, a company in Maconomy is given a certain access level. A user only has access to data that is related to that company, if the user has been granted access to the access level of the company or a higher access level. Otherwise, the user cannot find, browse, or see information that is related the company in question.

  For instance, when entering time sheet lines, you can only specify jobs in the **Job No.** field to which you have access, and if you use the Find menu to find jobs when entering a new time sheet line, the Find window only shows jobs to which you have access (and therefore can specify).

  Access levels are defined in the Access Levels window, and are assigned to users in the User Access Levels window. The individual access levels are assigned to data items in the various card windows, such as the Company Information Card.

- Direct and indirect access control — If an access level is specified for an object in Maconomy, this object is said to be under direct access control. For instance, for a job in the Jobs window, the access level that is specified for the job determines which users have access to it. However, there are many objects in Maconomy to which you cannot apply direct access control. For instance, a sales order or a time sheet does not contain a field for access level specification. Instead, access to such objects can depend on the user's access to one or several of the objects to which you refer. This way, access to a job entry, for example, can depend on whether the job entry contains a reference to a job to which you have access.

  For a full description of the Maconomy access control system, see the following section in this chapter, the introduction to the "Set-Up" chapter in the Maconomy Reference Manual, and the description of the windows mentioned in the preceding summary. The

introduction to the “Set-Up” chapter also describes examples of access control in the multi-company model and in the Job Cost module.

The chapter “Database Description” in the Maconomy Reference Manual contains a list of the relations for which direct access control can be specified.

The Maconomy Access Control System

This chapter contains an in-depth description of the Maconomy access control system.

The purpose of the access control system in Maconomy is to ensure that only information to which a given user has been granted access is accessible when using Maconomy. This applies to dialogs, search windows, and access through third-party products applying ODBC.

Specification of Access Principles

To make sure that the access principles are enforced, all reading of information in the Maconomy database is done through views, which are defined partly on the basis of the relation and partly on the basis of a description of the access principles that apply to the individual relations. The description of this information can be found in the database folder for the installation in question.

```
c:\maconomy\ <Maconomy-Home>\MaconomyDir\Database
```

Among many others, this folder contains the RelationDefinitions and RelationAccessDefinitions files. The RelationDefinitions file contains a definition of all relations in the Maconomy database. Similarly, the RelationAccessDefinitions file describes the access principles that are associated with the individual relations.

The access principles for each relation are structured as logic expressions, providing you with the possibility of assigning one or several expressions to the same relation. The available access principles are:

- NoAccessControl
- DirectAccessControl
- IndirectAccessControl

These access principles can be defined as follows.

NoAccessControl

This principle specifies that there should be no access control on the given relation. Any user can see the information stored in the relation.

DirectAccessControl

This principle specifies that the given relation contains a field with the name **AccessLevelName**, which relates directly to the definitions of access levels in Maconomy. Access to a given entry in a relation with this access control principle is granted if the user has access to the access level specified in the entry in question. For more information, see the introduction to the “SetUp” chapter in the Maconomy Reference Manual.

IndirectAccessControl

This principle specifies that access to the given relation is dependent on the access to another relation. For instance, access to journals in the G/L module could be dependent on whether the user has been granted access to the company that “owns” the journal. This example would be expressed as follows:

```
Journal : IndirectAccessControl (CompanyInformation, CompanyNumber)
```

The preceding line in effect says that “access to a given journal is dependent on whether the user has access to the company in the relation CompanyInformation, which has the CompanyNumber specified on the journal as key.”

As mentioned previously, the access principles are structured as logic expressions, which provides the possibility of specifying more than one condition. One example could be that to gain access to a requisition, a user needs access to the company that “owns” the requisition and access to the job to which the requisition is assigned. This example can be expressed as follows:

```
RequisitionHeader : IndirectAccessControl (CompanyInformation, CompanyNumber)
   and IndirectAccessControl (JobHeader, JobNumber)
```

As a final example, consider that there are no access principles on the relation InterestTableHeader. The access principle therefore looks as follows:

```
InterestTableHeader : NoAccessControl
```

Note that it is not possible to have indirect access control with a reference to a relation for which no access control has been specified. The result of this would be that there are no access restrictions on any of the relations in question. However, if you do this, you ensure that if, at some later point, access control is assigned to the relation to which the reference is made, access control will also be assigned to the relations whose access principles refer to this.

The general rule is that expressions that specify the access control for access to a given relation can consist of one or several elements separated by “and” or “or.” In addition, you can use parentheses, which make it possible to specify expressions of the following type:

```
DirectAccessControl and
(IndirectAccessControl (Customer, CustomerNumber) or IndirectAccessControl
(JobHeader, JobNumber) )
```

Changing Access Principles

To provide you with the possibility of making special adjustments to the access control system, a special set of programs has been designed to help you with the technical details.

Customer-Specific Access Principle File

When upgrading to new Maconomy versions, the RelationDefinitions and RelationAccessDefinitions files are overwritten with Maconomy’s standard access principles. To ensure that previous manual adjustments to the access control system are kept, you can create a file that comprises exceptions to the standard principles.

The name and location of this file is optional, but it is a good idea to place the file in the same folder as the original file that is supplied with Maconomy and use the following format for its name:

```
RelationAccessDefinitions.<CompanyName>
```

where `<CompanyName>` is the name of your company.

In this file, you can specify the access principles for the relations that should not use the standard principles. Thus it is the access principles in this file that will be used for the relations in question. The file must be a standard text file. When this specific access principle file is ready, the changes can be implemented as described in “Update of Access Principles” later in this document.

The following sections show some examples of how to adjust the access principles using a specific access principles file.

Example 1

Consider the access principle for the Creditor relation. Here the standard principle is DirectAccessControl, which means that the value of the **AccessLevelName** field for the individual vendor determines which user groups have access to the vendor in question.

You may want to change this principle so that all users have free access to all vendors. The access principle should therefore be changed from DirectAccessControl to NoAcccessControl.

In the specific access principle file you should specify the following line:

```
Creditor : NoAccessControl ;
```

Note the ; which indicates the end of the principle.

The adjustment can then be implemented as described in “Update of Access Principles.”

Example 2

Now look at the access principle for the relation JobHeader. Here the standard access principle is DirectAccessControl, which means that the value of the **AccessLevelName** field for the individual job determines which user groups have access to the job in question.

You may want to change this so that the principle also specifies that the customer must have access to the customer indicated on the job header and saved in the **CustomerNumber** field. The access principle should therefore be changed from DirectAccessControl to an expression that also includes the indirect access principle for the customer who is assigned to the job.

In the specific access principle file you should specify the following line:

```
JobHeader : DirectAccessControl
and IndirectAccessControl (Customer, CustomerNumber) ;
```

Note the ; which indicates the end of the principle.

The adjustment can then be implemented as described in “Update of Access Principles.”

Update of Access Principles

Access principles are updated in three steps:

1. Generate new view definitions
2. Delete existing views
3. Create views

It is essential that no users are logged in to the system during an update.

Generate New Views

To generate new views, the current folder must be set to the following:

```
<MaconomyHome>\MaconomyDir\Database
```

Now enter the following:

```
OracleViewGenerator-Kernel US -Administrator Administrator
-CheckOnly
-UseDD
-MaconomyFolder Folder "<DestinationFolder>" "RelationAccessDefinitions"
"RelationAccessDefinitions.<CompanyName>"
```

where "<DestinationFolder>" is the destination folder of the generated sql files. Use "." for the current folder.

For Maconomy version 6.0 and earlier, enter:

OracleViewGenerator –Kernel US –Administrator Administrator -AllWindows “All Windows” –CheckOnly “<DestinationFolder>” “RelationDefinitions” “RelationAccessdefinitions” “RelationAccessDefinitions<Customer>”.

Note that all parameters must be specified on the same line. In the above example, the parameters are entered on separate lines for the sake of clarity. The -CheckOnly parameter must be left out when generating a view, but must be included to validate whether all access principles are OK.

The folder path specified in the -MaconomyFolder option is the full path of the folder that contains your current MaconomyDir (placed in the MaconomyHomes folder).

In all cases you should use an OracleViewGenerator version 2.4 or later because of corrections in the views that are generated.

Delete Existing Views

The following two commands are used:

```
DeleteDHViews <shortname> <password> [MSQL] Delete3PStuff <shortname>
<password> [MSQL]
```

For Maconomy version 6.0 and below, enter:

```
OracleViewGenerator -Kernel US -Administrator Administrator
-AllWindows "All Windows" -CheckOnly "<DestinationFolder>"
"RelationDefinitions" "RelationAccessDefinitions"
"RelationAccessDefinitions.<Customer>".
```

Create Views

You can use one of the following commands—the first is for Oracle-based servers, the second is for SQL Server-based servers:

```
StartOracleSQL <shortname> <password> DHViewMake.sql StartSQLServer1
<shortname> <password> DHISQLViewMake.sql
MaconomyServer[MSQL].exe -iMaconomyServer.<Application> -f -3
-S<shortname>
Recreate3PStuff <shortname> <password> [MSQL]
```

Access Control in Servers and Clients

This section contains a short description of the implementation of extended access control in Maconomy’s server and clients.

General database views offer the possibility of implementing general access control; that is, the Technology layer enforces the access control by using the views. The extended access control is implemented in the following way.

In all information maintenance dialogs (such as Customer Information Card, Company Information Card, and so on), extended access control is used on all database lookups in the upper tab. Entries in the sub-tab are not included by the extended access control unless specified by the Application Department.

Lookups in the database in connection with printouts are not included by the extended access control unless specified by the Application Department.

All searches and exports from the File menu are included by the extended access control.

Lookups in the database from RGL reports are included by the extended access control if one of the expressions DirectAccessControl or IndirectAccessControl is included in the where clause.

When developing Analyzer reports, you can use the model for extended access control by using relation names prefixed by “AC” instead of “VW.” DirectAccessControl and IndirectAccessControl expressions in the where clause are subsequently subject to the same conditions as the Maconomy client.

The reason for distinguishing between views used in Analyzer reports and views used by third-party products is that Analyzer reports are run from the Maconomy client and hence they use another connection to the database. Unlike the client, third-party products need a connection to the database, which does not use the client’s normal login credentials. Therefore, VW views must be used from third-party products.

Because the extended access principles cannot be evaluated by the application, an automatic control of all foreign key references is built into the system. This control ensures, for example, that a customer number in the Order header refers to an existing customer, and that it can be used by the user according to RelationAccessDefinitions.

Secure Sockets Layer

The secure sockets layer (SSL) provides endpoint authentication and communications privacy over the Internet using cryptography. The protocols allow client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. In the case of Maconomy, a proper implementation of SSL means that nobody can “listen in” on the network traffic between the server and clients and thus pick up valuable business data.

SSL has three main functions:

- Authenticating the client and server to each other: the SSL protocol supports the use of standard key cryptographic techniques (public key encryption) to authenticate the communicating parties to each other.
- Ensuring data integrity: during a session, data cannot be either intentionally or unintentionally tampered with.
- Securing data privacy: data in transport between the client and the server must be protected from interception and be readable only by the intended recipient.

These objectives are met by having the server and the client exchange “handshakes” through a series of protocols and mechanisms that establish a secure channel between the server and the authenticated client. This channel is established every time that the client connects to the server.

Considerations

As with so many other areas, the increased security of SSL does come with a price: the added security layer has a performance premium. It is important to consider the tradeoff between security and performance carefully.

Performance

Every time that an SSL-encrypted connection is established between the server and a client, the SSL mechanism is invoked to verify the authenticity of the client and encrypt the data that is transferred. The performance overhead of the encryption part is negligible—less than 1%—however, establishing the secure connection involves a certain amount of network traffic and thus takes time.

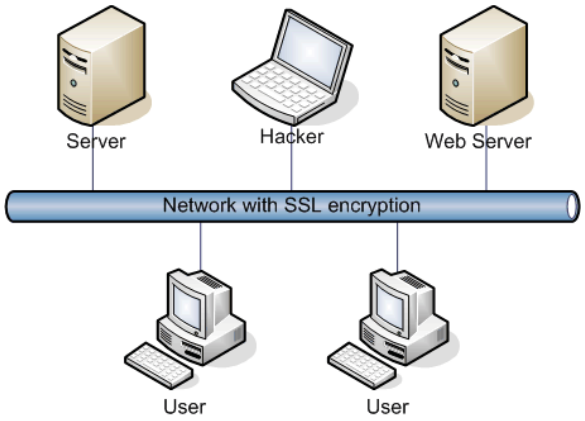
For web clients (the Portal and the Java Client), because of the nature of web protocols, a new connection is established and destroyed every time that data is sent to or received from the client.

Hence, a secure connection must be established every time, and that does carry a performance price.

Using Web Clients without SSL

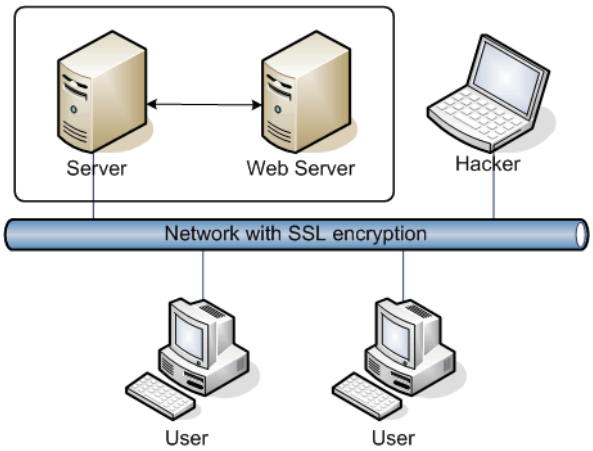
A main focus of the SSL protocol is to prevent outsiders—or insiders—from connecting to the network and “listening in” on the network traffic (this is called eavesdropping). To do this, a hacker needs physical access to the network wiring (or the wireless network broadcast—but that is an entirely different issue). Therefore, for maximum security, SSL should be enabled on all lines that are exposed to humans.

For Maconomy systems with web client (Portal/Java) users, this is no problem: Just enable SSL. Security is then increased without the network suffering a mentionable performance hit. The following drawing illustrates this. A potential hacker is symbolized by the laptop computer. His attempts at hacking the network are made difficult due to the SSL encryption.



The network is as safe as the Windows-only network in this diagram, but the Portal users experience performance issues. To resolve this, you can disable SSL for web clients only (see “Disabling SSL Clients”). However, in the preceding illustration this would mean that the network traffic between the Maconomy server and the web server is unencrypted, and the hacker could listen to the traffic.

The solution to this issue is the following topology:



Connect the web server directly to the Maconomy server, using a dedicated network interface card. Also make sure that the servers are located in a secure room. This way, a hacker cannot gain access to the network traffic between the Maconomy server and the web server.

Another solution would be to keep the web server and the Maconomy server on the same physical machine, but this raises other performance issues.

Enabling and Disabling SSL

SSL is enabled on the network by default. You can adjust this behavior by making entries in a custom version of the Maconomy.ini file on the Maconomy server.

Disabling SSL

Because SSL has been noted to cause problems on some platforms, notably Sun Solaris servers with continuous high loads, you can disable SSL altogether on the server by setting the following option in a custom version of the Maconomy.ini file:

```
[ServerConfig]
DisableSSL=true
```

You can also disable SSL for certain client types only, by listing the names of these client types:

```
[ServerConfig]
DisableSSL=<client-type> [,<client-type>]*
```

Currently there is only one supported client type for which SSL can be disabled: WebClient.

General Password Security

This section describes some guidelines for protecting your system using strong passwords. There is a lot if information about this subject on the Internet, but the following provides a number of Dos and Don'ts for you.

Default Password

When you install Maconomy, a default user called "Administrator" is created. This user has access to everything in the system. By default, the password for the Administrator user is "123456."

In the same way, every time that a user is created, the password is set to "123456." However, for ordinary users Maconomy automatically sets the user password as expired. This means that the first time that a user logs in to the system, he or she is forced to set up a new password in the Change Password window. For more information, see the "Setup" chapter in the Maconomy Reference Manual.

Rule number one is to make sure that no password in the Maconomy system is kept as a standard password ("123456"). This especially goes for the Administrator user.

This is especially important if you run the Maconomy Portal outside a firewall. It is possible to do a Google search for "Maconomy Portal" and click the search results to go to a Maconomy Portal login window. That login window could be yours—so make sure to use proper passwords.

The importance of passwords and how to create a proper password are explained further in the following.

Maintaining a High Level of Security

With external access to web portals, mail servers, and various workstations through the Internet and direct dial-in modems, a door for possible unwanted visitors is opened. The only way for an intruder to access the system is by guessing the correct combination of user ID and password. The main key to secure systems, therefore, is the correct use of passwords.

It is getting increasingly easy for an intruder to guess simple combinations of ID and password.

Do not underestimate the ease with which a password can be guessed or stolen. There are many techniques available to do this. Another simple and amazingly successful technique for the cracker is password-guessing.

It is up to every individual to ensure that his or her passwords are safe—a single unsafe password can open the door for a computer hacker to violate the integrity of the computer network or to steal information from your databases.

Maconomy supports a number of different password policies (length, use of special characters, restricted reuse of passwords, and periodic expiration dates). These policies are described by the following example.

Choosing a Safe Password

As discussed previously, a secure password is still one of the most important methods for protecting the network and servers from unwanted visitors.

The following simple guidelines are used in the industry to select and manage good passwords.

Do Not

- Use simple passwords that are easy to guess. Such as “123456” or “ppu123.”
- Use any word from a dictionary (of any language), because most forms of password attacks use dictionaries as a basis for password-guessing.
- Use user IDs (such as NFA), birthdays, car registration numbers, room numbers, department names, server names, locations, names of wife, husband, children, pets, and so on. Such names can be guessed, because most of this information is not confidential.
- Use the same password on multiple accounts. If you have many accounts, do not use the same password on each account. If one is broken, then all are broken.
- Reuse old passwords.
- Write your password down or save your password on a disk (for example, in an MS Word document).
- Tell or share your password with anyone, by e-mail or by any other means. Note that a PIN for a credit card can be compared to a password. If someone has the password, he or she can commit criminal acts using the credit card account.

Do

- Use a password that consists of at least 6 characters.
- Use mixed-case alphabetic characters.
- Include special characters—that is, nonalphabetic characters. Try to place them in the middle of the password.
- Change passwords frequently (for example, every 75th day).

General Techniques for Generating Safe Passwords

General techniques for generating safe passwords include the following:

- Take the first letter from each word of a phrase, sentence, theorem, poem, and so on (a pass phrase).
- Add some special characters—nonalphabetic, nonnumeric, punctuation, and so on.
- Deliberately misspell.

- Use the first letter from each word in an easy-to-remember sentence.

Example

Three Simple Steps to an Acceptable Password

1. Initial pass phrase: “Go ahead punk, make my day” becomes “gapmmd.”
2. Use capital letters. Convert some letters to uppercase: “GaPMmD.”
3. Insert some special characters: “GaP-MmD!”

Do **not** use this example, given that it has been published.

If you follow these guidelines, guessing your password should take longer than your password change frequency.

Reset Password

The Workspace Client features an integrated password reset process through a user can reset a forgotten password without having to contact the local Maconomy administrator.

Before You Begin

The password reset process is based on temporary authentication tokens sent to the user via email. Before you begin, ensure that a valid email address is specified for each user on the Employees pane.

Set Up Password Reset

To enable the password reset functionality in the Workspace Client for employees, administrators must complete the following steps to configure the system appropriately.

To enable password reset, complete the following steps:

1. Select the field **Enable Reset Password** on the OSGi products page in MConfig.
2. Configure the **MailPortAndServer** system parameter in the Maconomy application with the SMTP server name and port number to use for sending emails.
3. Ensure that all relevant users have valid email addresses specified in the **E-mail** field, located under Contact Information on their Employee card.

Additional Configuration Steps

Set Template with Local Specifications

The template that is used for the email that contains the temporary authentication token is by default read from the file **email_templates/passwordreset.txt** within the Coupling Service installation directory. By default this file is a plain text message, but it can be changed to include a Multipart/MIME message with both HTML and plain text versions.

Localized versions of the template can be added with the naming convention:

passwordreset_<locale>.txt

Where <country> and <locale> are standard locale specifications in the form of either a country (“en,” “fr,” and “da,” for example) or a country and language variant (“en_US,” “fr_FR,” and “da_DK”).

For example:

passwordreset_en_US.txt

Set “Sender” Email Address

The email address that appears as the “sender” of the authentication token emails is specified in the **server.auth.email-token.message.address** setting in the **configuration/server.ini** file. By default this address is **donotreply@deltek.com**. You can change the location and name of the email template with the **server.auth.email-token.message.template** setting.

Restrictions and Limitations

Due to security concerns, the reset password process does not report errors and configuration issues to the user. Instead it may state that it has sent an email, even though no email has been sent. If it appears that authentication token emails are not sent, the recommended diagnostic steps are as follows:

- Verify that a valid email address has been entered for the user.
- Verify the SMTP server name and port.
- Check that the specified server name is reachable from the computer that hosts the Coupling Service. Perhaps try to replace the server name with the resolved IP address.
- Enable debug logging for the email token authentication scheme in the Coupling Service by adding the following logger declaration to the **configuration/logback.xml** file:

```
<logger name="com.maconomy.server.auth.extensions.McEmailTokenAuthentication">
   <level value="DEBUG" />
   <appender-ref ref="FILE" />
 </logger>
```

- Note that the logging output is written to the **log/maconomy.log** file and may contain helpful information.

Use Password Reset

To use the password reset process, complete the following steps:

1. Select the **Forgot password?** link in the Workspace Client login window.
2. Enter your Maconomy user name.

   A temporary authentication token is generated and sent via email to the email address specified on your Employees card. The Workspace Client then prompts you to enter the token received by you via email.
3. Enter the token at the prompt. If the token is valid, a Reset Password dialog is displayed.
4. Enter and verify a new password. After a new valid password is entered and confirmed, you are automatically logged into the Workspace Client.

Single Sign-On

The Single Sign On (SSO) add-on to Maconomy allows users on Microsoft networks to let their passwords to the Microsoft network double as their passwords to the Portal.

Having to log in to Maconomy every time that the Portal is opened can be annoying for users. With SSO enabled, the Maconomy Portal does not display the login screen when started; instead, it opens the current user's Portal right away.

SSO can be set up to allow immediate access to the Portal in two ways. With name matching, a check is performed to see whether the user's network domain is allowed access to the portal by comparing it against a list of domain names that is configured on the Maconomy server. The user's network username is then looked up in Maconomy; if the user name exists, the user is logged in automatically. With name mapping, the user's name and domain are checked directly against entries in the Maconomy database. If a user with the current network user name and domain exists in Maconomy, the user is logged in to the Portal automatically.

If an error occurs, for example, if the domain is invalid or if the user is blocked or does not exist, the user is directed to the standard Portal login screen, where he or she must enter his or her Maconomy username and password.

Requirements

SSO builds on Microsoft technology. The following requirements exist:

- You must be running a Microsoft network.
- The web server must be Microsoft IIS 5.0 or later.
- The browser (clients) must be Microsoft Internet Explorer version 5.5 or later.
- The Portal must build on Portal Framework 4.0 or later.
- Maconomy TPU number 39 or later must be installed.

If appropriate, these limitations can be overcome by using SSO with an external verifier such as Kerberos, which was introduced in TPU 9.0. For more information, see "Single Sign On With Kerberos."

Bypassing SSO

If you do not want to use SSO (for example, to provide the Portal for users on other network platforms), the SSO procedure can be bypassed by adding the query variable SSO=no to the URL engaging the portal. If, for instance, the standard URL is

```
http://...../MaconomyPortal.exe/maconomyportal.msc
```

then the URL

```
http://...../MaconomyPortal.exe/maconomyportal.msc?SSO=no
```

will always prompt users for username and password ("old-style" login procedure).

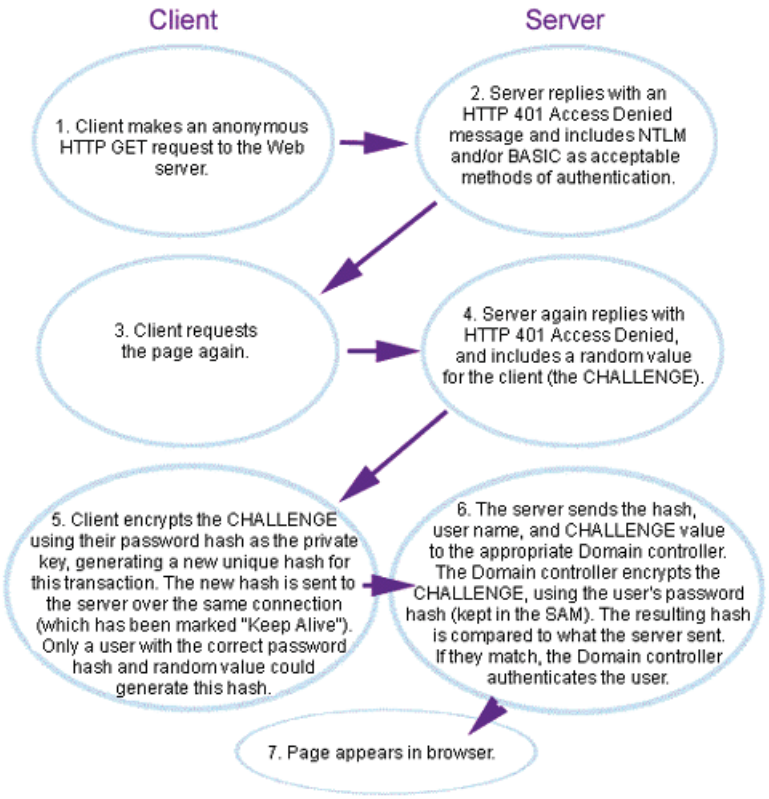
Configuration

To configure the web server and Maconomy system to use SSO, please follow the guidelines that are described in this section.

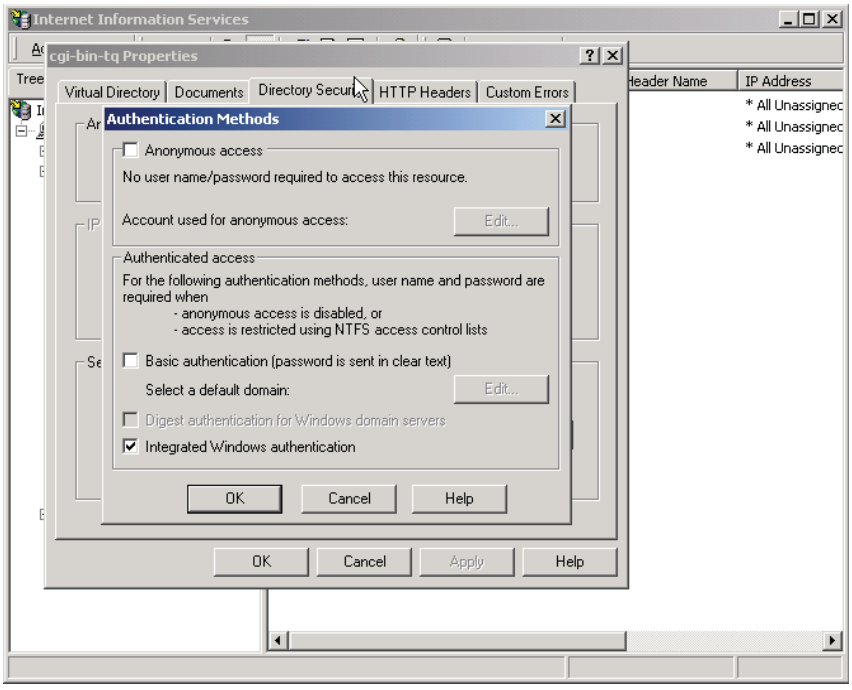
Web Server

Maconomy (in the form of M-Script) uses the Microsoft Challenge Response protocol to automatically detect the username and domain of the current process. The Challenge Response protocol automatically verifies the username and domain from a web browser connecting to a web server.

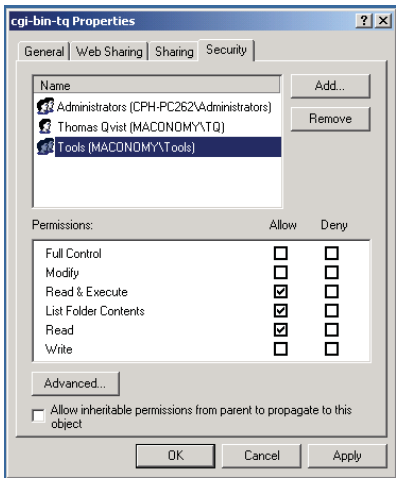
The protocol used between browser and web server is illustrated in the following figure (courtesy of Microsoft's website):



To set up the web server to verify users, IIS should be configured to only allow authenticated users to access the CGI directory in which the M-Script executable resides.



In addition, the actual access to the CGI directory should be limited to the users and access groups that are allowed to access the web site. The following figure shows how the standard security settings for the CGI directory should be set.



When using single sign on, the CGI program (MaconomyMscript.exe) is executed with the privileges of the browser user (see “An M-Script Login Session”). For this reason, some special security considerations must be made regarding the file system on the web server:

- Because the CGI program reads M-Script program files, the access rights of these files should allow all valid users read access.
- Because the CGI program writes session information to the sessions directory, all valid users should have both read and write access to this directory.

  This does not compromise the security of the web server. Access to the web server can be restricted in other ways so that no users actually see these files and directories.

This concludes the setup of the web server.

Maconomy Server

The Maconomy server must also be configured to use SSO. This is done in the file maconomy.ini, which is located in the definitions folder on the Maconomy server.

Four entries pertain to SSO:

1. `SSO=off | namematch | namemapping`

   This setting turns SSO on or off. The default value is off. SSO can be turned on using one of two methods: name matching or name mapping.

   With name matching, the network username must match the Maconomy username, and the option SSODomain must match a network domain name.

   With name mapping, the user name and domain name of the user are compared to the value in the “Network User Name” and “Network Domain Name” fields in the Users workspace in the Set-Up module in Maconomy. If the network user name does not find a matching entry in Maconomy, the login fails. If the network user name is matched by more than one Maconomy user, Maconomy selects a user at random. Note that the user name and domain name must be entered in capital letters in the Users workspace, because the actual user name and domain name that are supplied by the network are converted to uppercase before attempting the match against the user data in Maconomy.

2. `SSODomain=<Domain name list>`

   When using the name-matching SSO method, the users must exist in one of the domains in the domain name list, which is set using this parameter. This entry is mandatory when using the name-matching SSO method. The domain names are checked against the

domain name of the user who is trying to log in. The check is case-insensitive, so, for example, mydomain and MyDomain match.

This option should not be used if SSO=namemapping (see the previous step).

3. `SSOAllow=<address list>`

   See the following step.

4. `SSODeny=<address list>`

   The Allow and Deny lists limit the IP addresses of web servers from where SSO is allowed. It is recommended to only allow web servers that run the Maconomy Portal.

   <address list> has the following format:

   ```
   <address> [ '-' <address> ] { ';' <address> [ '-' <address> ] }
   ```

   Examples:

   ```
   SSODeny=10.22.33.44
   SSOAllow=127.0.0.1; 10.22.33.0 - 10.22.33.255
   ```

   The check algorithm is as follows:

   a. Look for the address in the Deny list. If it is found, reject the connection.

   b. Look for the address in the Allow list. If it is found, accept the connection.

   c. If the address is not found in either list, reject the connection.

   If no SSODeny or SSOAllow entry is found in Maconomy.ini, all connections are denied.

   Example of SSO-related entries in the `maconomy.ini` file:

   ```
   SSO=namematch
   SSODomain=MyDomain,OurDomain,AnotherDomain
   SSOAllow=10.22.33.1-10.22.33.255;127.0.01
   SSODeny=10.22.33.44
   ```

Add-On Number

To use SSO, the installation must be updated with an installation number and add-on set number 79.

M-Script Implications

To make use of the SSO functionality, the M-Script command `maconomy::singleLogin(...)` is used.

This command uses the Microsoft Challenge Response protocol to automatically detect the username and domain of the current process. This information is sent to the server, where SSO is attempted. If SSO fails, an exception is thrown. The M-Script program then typically redirects the user to a web page where he or she can perform a standard login.

Portal Framework

A setting in the file framework/conf/portal.I specifies whether the standard login script should attempt to perform an SSO before defaulting to the “old-style” login procedure:

```
singleLogin: true
```

The setting is in the file framework/conf/portal.l file. However, prior to that, this setting was configured in the framework/params.txt file.

In addition, the script tests for the occurrence of the query variable SSO. If this variable is set to false, SSO is not attempted. See “Bypassing SSO.”

Flow Summary

This section describes the flow of an M-Script login session when SSO is installed correctly.

An M-Script Login Session

1. When a user attempts to get an M-Script page from the web server, for example, when trying to access the Portal, the web browser and web server automatically negotiate. The M-Script program is only started if the browser can document that it is a valid user (according to the security settings set up for the CGI directory).

   This negotiation happens automatically (that is, single sign on is used) if the user is already logged in to his or her workstation as a valid user. If the user is not logged in as a valid user, he or she is prompted with a standard browser password dialog, where he or she should enter a valid username, password, and domain (note that Internet Explorer on the Macintosh also does this, because no valid login exists on the Macintosh client).

2. The CGI program (that is, MaconomyMscript.exe) is started. The web server starts the process as the browser user. This has some implications with respect to the security settings of the local file system on the web server, which is discussed in the previous “Web Server” section.

   The web server supplies the CGI program with information about the domain and username for the current user, as well as an NTLM authentication string.

   The CGI program then compares the domain and username with the actual username and domain name for the process. This makes it impossible to enter a false username by just changing the parameters from the web server to the CGI program.

3. The information is sent to the Maconomy server with a request for single sign on.

   The Maconomy server first checks that the IP address of the web server is within the SSOAllow list of web servers that allow single sign on, and not in the SSODeny list (see “Maconomy Server”). This excludes the possibility of forging a single sign on from another server.

4. Upon successful checks in step 3, the M-Script command (see “M-Script Implications”) succeeds, and information about the actual Maconomy username is returned to the CGI program. Upon failure, the M-Script command returns error information that corresponds to the error information that is returned upon a failed standard login.

The M-Script program then (upon failure) typically redirects the user to the standard login page. This page is also available through a URL, to allow login as a specific user without single sign on.

Single Sign On with Kerberos

The Single Sign On (SSO) add-on to Maconomy allows users to let their network password double as the password to the Maconomy client, including the Portal.

Having to log in every time that the Maconomy client or Portal is opened can be annoying for the users. With SSO enabled, Maconomy does not display the login screen when started, but opens right away.

SSO can be set up to allow immediate access in two ways. With name matching, a check is performed to see whether the user's network domain is allowed access to Maconomy by comparing it against a list of domain names that is configured at the Maconomy server. The user's network username is then looked up in Maconomy; if the user name exists, the user is logged in automatically. With name mapping, the user's name and domain are checked directly against entries in the Maconomy database. If a user with the current network user name and domain exists in Maconomy, the user is logged in to Maconomy automatically.

If an error occurs, for example, if the domain is invalid or the user is blocked or does not exist, the user is directed to the standard login screen, where he or she must enter his or her Maconomy username and password.

Previous SSO implementation

TPU 39 introduced a precursor to the present SSO implementation. That implementation had some drawbacks:

- It only worked for Portal clients.
- You had to be running a Microsoft network.
- The web server had to be Microsoft IIS.
- The browser clients had to be Microsoft Internet Explorer.

With the current implementation, which was introduced with Maconomy 9.0/TPU 9.0, single sign on to any Maconomy client except the native Macintosh client is supported, regardless of network, platform, or web server.

However, if the former implementation fulfills your requirements, it is still supported. For more information, see "Single Sign On."

Requirements

The following requirements exist:

- Add-on 79 (SSO).
- The authentication server must support Kerberos v5 with GSS (Generic Security Services) API.
- Sun or IBM Java version 1.4 or newer should be installed on the application server machine.
- Maconomy TPU 9.0 or later must be installed.

Bypassing SSO

If you do not want to use SSO—for example, to provide the Portal for users on network platforms for which SSO has not yet been set up)—the SSO procedure can be bypassed.

For the web client (the Portal), you can bypass SSO by adding the query variable SSO=no to the URL engaging the portal. If, for instance, the standard URL is

```
http://...../MaconomyPortal.exe/maconomyportal.msc
```

the URL

```
http://...../MaconomyPortal.exe/maconomyportal.msc?SSO=no
```

will always prompt users for username and password ("old-style" login procedure). You can, for example, use the supportedclients.I file to direct certain browser types to the non-SSO login (see the manual "Portal Configuration and Administration").

For Windows clients, you can enter SingleLogin=false in the client's maconomy.ini file, which is located in the Definitions subfolder of the Maconomy folder.

System Overview

This implementation of SSO is based on the *Kerberos* security protocol.

> "Kerberos is a distributed authentication service that allows a process (a client) running on behalf of a principal (a user) to prove its identity to a verifier (an application server, or just server) without sending data across the network that might allow an attacker or the verifier to subsequently impersonate the principal. Kerberos optionally provides integrity and confidentiality for data sent between the client and server."[1]

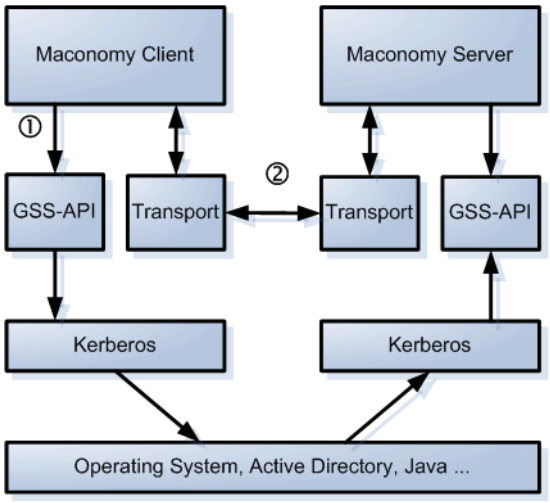
Kerberos was developed by the Massachusetts Institute of Technology (MIT). Maconomy uses an external verifier plugin based on the Kerberos security protocol v.1. The plugin is compatible with Kerberos-based authentication devices that implement the GSS-API (Generic Security Services API). This includes Microsoft Active Directory and the Linux platforms supported by Maconomy. This plugin also supports SSO from the Maconomy client for Windows (v. 4.6.0 or newer) and the Maconomy client for the Java™ platform (v. 3.7.0 or newer).

A later section in this document provides information about how to use a custom verifier for the Portal. Kerberos works by providing principals (users or services) with:

- tickets that principals can use to identify themselves to other principals, and
- secret cryptographic keys for secure communication with other principals.

A ticket is a sequence of a few hundred bytes. Kerberos provides for mutual authentication and secure communication between principals on an open network by manufacturing secret keys for any requestor and providing a mechanism for these secret keys to be safely sent through the network.

As mentioned previously, Maconomy implements the Kerberos functionality through a device called the GSS-API. The following diagram illustrates Maconomy's Kerberos implementation.



The initial communication (1 in the preceding diagram) from the Maconomy client to the Maconomy server is established through a sequence of communications between the GSS-API,

[1] B. Clifford Neuman,Theodore Ts'o: "Kerberos: An Authentication Service for Computer Networks." See for example http://www.isi.edu/gost/publications/kerberos-neuman-tso.html

Kerberos, the operating system (or Microsoft Active Directory or similar), and the Maconomy server. Through a system of tickets and encrypted security keys, Kerberos verifies that the Maconomy client is indeed the user that it claims to be, and that it is safe for the Maconomy server to establish communication with the client. After this initial handshake, the server and client can communicate using Maconomy's network transport protocols without involving Kerberos (2 in the preceding diagram).

Setup

This section describes how to set up Maconomy to use Single Sign On (SSO) using the Kerberos security protocol, by means of Java components that are included in the Maconomy TPU or by using the SPNEGO protocol.

The Portal can use a custom verifier (replacing Kerberos). This is described in a subsequent section.

Client Setup

Portal

To enable SSO in the Portal, you must add the following settings to the portal.I file on the web server:

```
singleLogin: true, // This is the default. If false, the Portal shows
                   // the standard login screen and does not attempt
                   // single sign on.

singleLoginBrowserMethod: "spnego", // This is the default, the SPNEGO protocol
                                    // is used. Other values can be "java"
                                    // if the SSO process should be done
                                    // using Java Applets, optional.

javaEnabled: true, // This is the default. If false then all Java
                   // components in the Maconomy Portal will be disabled,
                   // also: setting this to false implicitly disables
                   // Java Applet based SSO. Use this if disabling use
                   // of Java Applets in the Maconomy Portal, optional.

singleLoginMethod: "external", // If not specified, the old SSO method
                               // is attempted.
pageShownOnLogout: "URL",  // Optional. "URL" is a string starting
                           // with http:// or https://
```

A few comments related to some of the settings above:

"`singleLoginBrowserMethod`" defines how browsers implement SSO:

- "`java`" specifies Java Applet based SSO, which was previously the default (before Maconomy versions 2.1.8, 2.2.5, 2.3.1)
- "`spnego`" specifies the new SPNEGO based browser SSO. "`spnego`" is the default (in Maconomy versions 2.1.8, 2.2.5, 2.3.1 and newer)

"`javaEnabled`" defines if Java Applets are supported by the Maconomy Portal at all.

The Maconomy Portal will ask the user to install Java if it is not available.

Setting "`javaEnabled`" to "`false`" disables this behavior and disables any Java Applet components in the Maconomy Portal. Setting " `javaEnabled`" to "`false`" also disables Java Applet-based SSO and Java version checking.

The "`singleLoginBrowserMethod`" and "`javaEnabled`" options are available in Maconomy versions 2.1.8, 2.2.5, 2.3.1 and newer.

Please note that it's still possible to run Java Applet based Maconomy Portal components even if using SPNEGO browser based SSO, just ensure that "`javaEnabled`" is "`true`".

If you specify a URL with the optional setting **pageShownOnLogout**, the browser redirects to the specified URL when you log out of the Portal. In this way you can prevent users who do not get to see the login screen because of SSO from seeing the logout screen when they log out.

The Portal login script tests the URL for the occurrence of the query variable SSOCookie. If this variable is set, the script tries to find this cookie and extract its value. Note that the portal script only recognizes cookie values that contain the letters a-z, A-Z, numbers, and the following special characters: :_@().

Login script tests for the occurrence of the query variable SSO. If this variable is set to false, SSO is not attempted.

Browser

If using SPNEGO-based SSO, both PC and Mac browsers need to be set up to use SPNEGO-based SSO using Kerberos.

Browser Setup for Internet Explorer

For Internet Explorer (IE), you may need to add the Maconomy Portal address to the Local intranet zone, as IE does not permit Kerberos-based SPNEGO SSO for websites.

More details on using SPNEGO based SSO in Internet Explorer are available in the "Client Side—Internet Explorer" section of the following Microsoft article:

`https://msdn.microsoft.com/en-us/library/ms995329.aspx`

Browser Setup for Firefox on Windows and Mac OS X / macOS

For Firefox, you may need to set the property "network.negotiate-auth.trusted-uris" in the "about:config" page of FireFox.

1. Open the page **about:config**.Accept the warning that appears
2. In the **Search** field find the property **network.negotiate-auth.trusted-uris**.
3. Double-click **network.negotiate-auth.trusted-uris**
4. In the editor window that appears, enter a comma-separated list of http:// URLs on the Maconomy Portal site. Click **OK** to accept the changes.
5. Close the **about:config** page.

Multiple Realms and the Java Client

The following information is only relevant for Java clients that run on non-Windows platforms, and for Java clients where native SSO has been disabled.

If multiple realms have been defined on the server (see “Server Setup”), the Java client requires a standard Kerberos configuration file to perform SSO. The file is called krb5.conf, and it is generated automatically each time that the client starts up; it is present while the client is running, and it is normally deleted when the client exits (except when debugging is enabled, in which case the file is left after the client exits.) The file is placed in the Maconomy/Jaconomy folder in the application data directory of the local user (on most versions of Microsoft Windows, for example, this is the hidden folder <user-dir>/Application Data).

You can override the setting in the autogenerated configuration file, for example to perform local configuration experiments, by placing a second configuration file called krb5.local.conf in the same folder. If such a file is present, the Java client uses it, and the standard configuration file is not generated. A local configuration file is not, of course, deleted automatically.

Documentation of the krb4.conf file format and the set of properties that is supported can be found as part of the documentation of your Kerberos documentation, or at http://web.mit.edu/Kerberos/.

Active Directory and Java Security in the Portal, Java Client, and Workspace Client

As of Maconomy 2.1 all Maconomy clients use native Windows APIs to obtain a Kerberos ticket for the Maconomy installation. Before Maconomy 2.1, a change to the registry settings on each client machine was required for Kerberos Single Sign-On to work.

Information about the registry that needed to be set is included here for reference purposes only. It is no longer required.

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters`

Value Name: `allowtgtsessionkey`

Value Type: `REG_DWORD`

Value Data: `1`

Server Setup

For a Maconomy client to be verified by Kerberos, a Maconomy service must be registered as a Kerberos service. This means that a specific network user account should be assigned to Maconomy (for example, the maconomy user) and a Maconomy service should be registered to this account (making it a service account). For the verifier (Kerberos) to authenticate users, it must be logged in to this account. The login information is stored in a configuration file, which you must place on the server. The content of the configuration file should have the following form:

```
accountUsername=<account name> accountPassword=<account password>
```

In this file you can also specify whether the domain name for the user should be returned as part of the authentication response:

```
returnDomainName=<true/false>
```

By default, the domain name is not returned, and the option is only needed if the network domain name has also been specified for all users in the Maconomy Users card.

The path to the configuration file should be specified in the Maconomy.ini file as described in the following.

In addition, clients that connect to the server must know the name of the service and have information about the Kerberos realm (domain) that they want to log in to. This is configured by settings in the Maconomy.ini file on the server.

Note that special considerations apply to the AIX platform.

The configuration in the Maconomy.ini file is done in the following way. Numbers in the following listing refer to the annotations that appear after the listing:

```
  [ServerConfig]
1 SSO_ServiceName = <name>
2 SSO_KerberosKDC = <address>
3 SSO_KerberosRealm = <name>
4 SSO_KerberosRealms = <realm1>=<kdc1>,<realm2>=<kdc2>,...
5 SSO_ExternalVerifier = <path to Kerberos>
6 SingleLogin=WinClient,WebClient
  [ApplicationConfig]
7 SSO=namemapping
```

1. This is the name of the service Maconomy that has been registered as the authenticator, for example:

   ```
   SSO_ServiceName = Maconomy/test.example.com
   ```

2. This is the network name or IP address of the Kerberos KDC (Key Distribution Center).

3. This is the name of the realm (domain) under which the Maconomy service has been registered, for example:

   ```
   SSO_KerberosRealm = example.com
   ```

4. For multi-realm authentication to work, this property must be set to contain a list of all of the realms from which SSO should be possible, along with either the network name or the IP address of the associated KDC. If all SSO users reside on the same realm as the Maconomy service, this property does not have to be defined.

   In a multi-realm setup, all of the realms (domains) must be configured with a direct two-way trust relationship to the primary realm (the one that hosts the Maconomy service). It must be possible for the clients to "jump" directly from the local realm to the primary realm using the local Ticket Granting Ticket.

   For an Active Directory domain forest, a two-way trust is automatically established between each domain and its parent domain, but for other installations it can be necessary to establish the necessary trust relationship explicitly. See the documentation for the Kerberos installation that you are using for additional information.

5. Apart from this, the server must know the path to the external verifier (Kerberos). This is expressed in the following way:

   ```
   SSO_ExternalVerifier=<tpudir>/../jre/{platform}/{architecture}/bin/java
       -jar <tpudir>/KerberosSSOExternalVerifier.jar {inifilepath}
       <realm> <kdc> <ticket>
   ```

   Parameter placeholders enclosed in <…> are automatically expanded to their corresponding values by the server. The supported parameter placeholders are as follows:

   <tpudir>: The TPU path for the application

   <realm>: The value from the SSO_KerberosRealm configuration setting.

   <kdc>: The value from the SSO_KerberosKDC configuration setting.

   <ticket>: The ticket received from the client.

   The following parameter values must be supplied manually:

   {*platform*}/{*architecture*}: The OS and architecture of the TPU-bundled JRE (for example win32/x86).

   {*inifilepath*}: Specifies the path to and name of the SSO ini file (see above). If you place the file in the MaconomyDir for the applicable Maconomy application, you can use the abbreviation <maconomydir> instead of the full path name (for example <maconomydir>/SSO.ini).

6. To specify which clients are available for SSO:

   ```
   SingleLogin=<list>
   ```

   where <list> is either WinClient and/or WebClient, or All or None.

7. Finally, to activate SSO on the server, enter the following in the [ApplicationConfig] section of the Maconomy.ini file:

   ```
   SSO=off | namematch | namemapping
   ```

The default is off. SSO can be turned on using one of two methods: name matching or name mapping.

With name matching, the network username must match the Maconomy username.

With name mapping, the username of the user is compared to the value in the **Network User Name** field in the Users window in the Set-Up module in Maconomy. If the network username does not find a matching entry in Maconomy, the login fails. If the network username is matched by more than one Maconomy user, Maconomy selects a user at random. Note that the username must be entered in uppercase in the Users window, because the actual username that the Kerberos KDC supplies is converted to uppercase before attempting the match against the username in Maconomy.

Note that this is different from the functionality of the “old” SSO, in which the domain name was also checked against the **Network Domain Name** field. The reason for the change is that the domain name is (normally) always the same for the Kerberos SSO. However, you can configure the system to also check for the domain name by adding the following line to the SSO configuration file (the file that contains the account login information as described previously):

```
returnDomainName=true
```

Note that the domain name must also be entered in uppercase in the Users window for a match to succeed.

Web Server Setup

When using SPNEGO to implement SSO in the Maconomy Portal, certain web server preparations are required..

Apache

When using the Apache web server for SPNEGO SSO in the Maconomy Portal, ensure that the following changes are applied.

For Apache version 2.4.12 or older (in the `<virtualhost>` tag for the Maconomy Portal), add the following to the Apache “`httpd.conf`” file.

```
SetEnvIf Authorization "(..*)" HTTP_AUTHORIZATION=$1
```

Please note the “..” in the above snippet--this is needed.

This instructs the Apache web server to add the cgi-program environment variable “`HTTP_AUTHORIZATION`” if the HTTP header “`Authorization`” is present in a HTTP request. The “`HTTP_AUTHORIZATION`“ cgi-environment variable is used by the Maconony Portal to perform SPNEGO-based SSO.

Normally Apache version 2.4.12 and earlier supresses the “`HTTP_AUTHORIZATION`“ environment variable, but it can be recovered by adding the above snippet.

For Apache version 2.4.13 or newer (in the `<directory>` tag of the `<virtualhost>` tag for the Maconomy Portal), add the following:

```
CGIPassAuth On
```

This instructs Apache to pass on the HTTP authorization headers as cgi-program environment variables, e.g. "`HTTP_AUTHORIZATION`".

For more detailed information about `CGIPassAuth` see the following links:

- http://stackoverflow.com/questions/17018586/apache-2-4-php-fpm-and-authorization-headers
- https://httpd.apache.org/docs/2.4/en/mod/core.html#cgipassauth

Internet Information Server (IIS)

When using the IIS web server for SPNEGO SSO in the Maconomy Portal,

1. Ensure that Windows Authentication has been disabled for the following folders:
   - cgi-bin
   - `cgi-bin/maconomy`
   - `maconomy`

If Windows Authentication is not disabled for these folders, IIS interferes with the Maconomy Portal when using the SPNEGO protocol. This occurs because IIS itself attempts to do Windows Authentication using the SPNEGO protocol.

2. Create a `web.config` file in the `cgi-bin/maconomy` folder with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <httpErrors existingResponse="PassThrough" />
    </system.webServer>
</configuration>
```

3. Associate the `web.config` file with IIS using the "Management -> Configuration Editor" icon in the IIS configuration page for the "`cgi-bin/maconomy`" folder.

   This prevents IIS from taking over and handling the `HTTP 401` result code from the Maconomy Portal and return an "error page" instead of the HTML from the Maconomy Portal cgi-programs.

The Maconomy Portal itself has a special `HTTP 401` "error-page" that redirects the browser back to the Maconomy login page. This is needed if the browser is unable to participate in a `HTTP 401` / SPNEGO login (e.g. if the PC or Mac is not on a (known) domain), then some browsers instead choose to show the response of the failed `HTTP 401` request.

Active Directory Setup

If using SPNEGO for SSO, the Maconomy Portal now requires that the Maconomy Portal Web site is made known to the AD/Kerberos server using the `setspn` command.

The guidelines for this is just like the guidelines for SPNEGO based SSO for iAccess and Touch.

On the AD/Kerberos server, run the following commands to make the Maconomy Portal host known as a Kerberos HTTP service:

```
setspn -a HTTP/<host> <account>
setspn -a HTTP/<host>.<domain> <account>
```

Example

The following is an example of how SSO can be set up with Maconomy in Microsoft Active Directory (MSAD). Note that this example might not be applicable for all versions of MSAD, and that other deployment methods are possible. See the documentation that is supplied with your version of MSAD.

In the following it is assumed that an MSAD has been set up on the domain (realm) called example.com. The MSAD is assumed to be installed on a computer that has the network ID test.

Implementation Procedure

To set SSO up with Maconomy in Microsoft Active Directory, complete the following steps:

1. Create or update a user account on the domain. In the following it is assumed that the account has the name Maconomy, and the password for the account is password.
2. Depending on your security policies, you may want to select the following Account options:
   - Account is trusted for delegation
   - Account is sensitive and cannot be delegated
   - User cannot change password
   - Password never expires
3. Assign a Service Principal Name (SPN) to the account. This is done using the SETSPN.EXE tool, which is part of the System Support Tools for the operating system. Running the following command creates an SPN called **macoservice/test.example.com** for the Maconomy user:

   ```
   setspn.exe –A macoservice/test.example.com Maconomy
   ```

   If using SPNEGO-based SSO, assign a SPN to the Maconomy Portal web-server host to the account. This uses the SETSPN.EXE tool, which is part of the System Support Tools for the operating system. Running the following command creates an SPN called **HTTP/test.example.com** for the Maconomy user.:

   ```
   setspn.exe –A HTTP/test.example.com Maconomy
   ```

   In the **MaconomyDir** folder for the application on the server, create a file named **SSO**.I

(you can give it another name, as long as you reference it correctly in Maconomy.ini—see the following). The file should contain the following settings:

```
accountUsername=Maconomy
accountPassword=password
```

4. In the Maconomy.ini file on the server, add the following:

```
[ServerConfig]
…
SSO_ExternalVerifier=java -jar
<tpudir>\KerberosSSOExternalVerifier.jar <maonomydir>\SSO.I
<realm> <kdc> <ticket>
SSO_KerberosRealm=example.com
SSO_KerberosKDC=test
SSO_ServiceName=macoservice/test.example.com
SingleLogin=WebClient,WinClient
```

5. Add the following lines to the Portal.I file used by the Portal on the web server (if the singleLogin property is already there, it should merely be updated):

```
singleLogin:              true,
singleLoginMethod:        "external",
singleLoginBrowserMethod: "spnego",
javaEnabled:              false
```

   The "`singleLoginBrowserMethod`" and "`javaEnabled`" properties are only available in Maconomy versions 2.1.8, 2.2.5, 2.3.1 and newer.

6. If the Maconomy usernames equal the network usernames, add the following line to Maconomy.ini:

```
[ApplicationConfig]
…
SSO=namematch
…
```

   Otherwise, you should add the following line and update user information for users with their network username in the Maconomy client as described previously:

```
[ApplicationConfig]
…
SSO=namemapping
…
```

7. If using the Apache web server and SPNEGO based SSO, then add the "`SetEnvIf Authorization "(..*)" HTTP_AUTHORIZATION=$1`" or "`CGIPassAuth On`" to the "httpd.conf" file for the Apache web server as described previously.

8. If using the IIS web server and SPNEGO based SSO, then add do the modifications for Windows Authentication and add the "`web.config`" file as described previously.

The Maconomy system should now be set up for Single Sign On for all users on example.com running the Portal or the Maconomy client for Windows or the Java platform.

Special Notes about AIX

Before the release of 9.0 service pack 6, SSO was not supported on AIX because no Sun JRE implementation exists for AIX. However, as of 9.0 SP6, the external verifier plugin released with Maconomy can also use the IBM JRE (version 1.4.2 or later).

While testing SSO on AIX, Deltek experienced some problems when using the service name with which Maconomy had been registered in the Maconomy.ini file, for example:

```
SSO_ServiceName = maconomy/test.example.com
```

It seems that the IBM verification implementation requires that the service name matches the account name of the authenticator—that is, the account name with which the verifier logs in. Thus, when the verifier logs in as Maconomy, and the service name is maconomy/test.example.com, authentication is denied.

However, changing the service name to match the account name instead:

```
SSO_ServiceName = maconomy
```

caused SSO to work. Note also that the case of the account name must match that of the service name.

Debugging

While testing the SSO setup, it can be helpful to log debug information. Because the messages from the Kerberos and GSS layers are not always very helpful, you can add a debug option to the external verifier.

To enable debugging, add the following option to the external verifier configuration file (typically named SSO.I):

```
debug=true
```

The debug output is logged in a file called KerberosServerPlugin.log in the Maconomy tmp directory. Note that the log file is overwritten each time that the plugin is called.

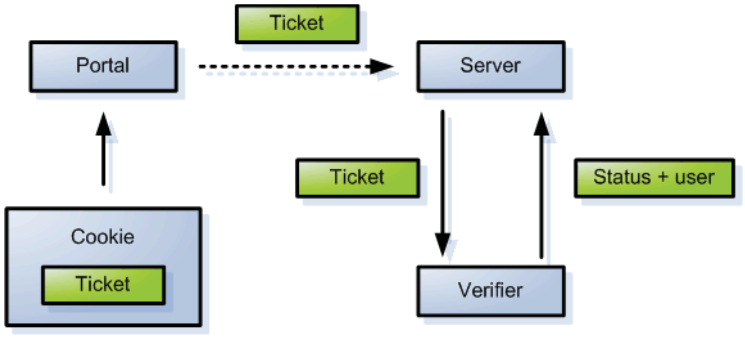
SSO on the Macintosh

The Portal and the Maconomy client for the Java™ platform can run with Kerberos SSO on Macintosh computers that run OS X 10.4 (Tiger). See http://web.mit.edu/macdev/KfM/Common/Documentation/preferences-osx.html for more information about setting up Kerberos SSO on a Macintosh.

Using a Custom Verifier

If you want to use a custom verifier—that is, an alternative to Kerberos—you can do so for the Portal users.

The procedure can be illustrated as follows.



1. The portal framework receives a cookie that contains a ticket in the form of a string.
2. The portal framework calls the server with the ticket string.
3. The server calls the external verifier with the ticket as argument.
4. The verifier returns a status code and a user name or an error message, depending on whether the ticket is valid for login.
5. If the verifier authenticated the ticket, the server logs the user in.

To set this up, you must do the following.

Portal

Set up the portal as mentioned previously by editing the portal.l file to specify that the standard login script should attempt to perform an SSO before defaulting to the “old- style” login procedure (see “Client Setup”).

Server

Edit the Maconomy.ini file as mentioned in “Server Setup,” but change the path for the setting SSO_ExternalVerifier to your custom verifier executable, for instance:

```
SSO_ExternalVerifier=C:\Program Files\MyVerifier.exe <ticket>
```

Maconomy automatically replaces the <ticket> argument with the value of the ticket in question.

External Verifier

The external verifier should be some form of executable, which takes the ticket as (one of) its command-line arguments. The result of the verification should be written to stdout and must be in the form:

```
<integer> , <string> [ , <string>] ;
```

where the integer is an error code, which is interpreted as follows:

```
<0 : internal error in verifier
0      : login succeeded.
: login denied.
: ticket information not sufficient for login other: internal error resulting
in a server crash.
```

If the error code is 0, the server expects it to be followed by a user name and probably a domain name. Otherwise, it expects it to be followed by an error message.

Examples of valid output strings:

```
"0,Jack Johnson;"
"0,Jack Johnson,example.com;" "-1, Unable to parse ticket;" "1, User not authorized;"
```

Font Administration

This document describes the use of import programs for Maconomy 2.0+. Before you proceed, make sure that your import program and program-specific import manual match the same Maconomy version and service pack number as this document, as a mismatch may cause errors in your import. This document describes the central concepts involved in the import process and explains the possibilities offered by import programs. Furthermore, this manual sets up a number of examples for the practical use of import programs. The concept definitions and examples are described on a general level. See the *Getting Started* for an overview of working with a specific import program.

Please note that the functionality described in this manual only applies to those Maconomy import programs that are generated according to the Standard Import Format. Maconomy's product range also includes a number of import programs with functionality that is not covered in this manual. You should therefore check the manual for the import program in question to see whether the program was generated using Maconomy's Standard Import Format.

Introduction

This document describes the use of fonts in Maconomy. This document is divided into six sections. The first four sections are relevant for MPL 2, and the last two are relevant for MPL 4.

1. Administration and installation of Type 1 fonts, including how you can make Maconomy aware of the installed fonts.
2. The Maconomy PDF printer driver, PrintToPDF.exe, including how you can convert Maconomy print files into properly formatted PDF files.
3. Using Type 1 fonts with MPL and MDL.
4. Example: Installing a font in Maconomy.
5. Administration and installation of fonts in MPL 4.
6. Example: Installing a font in MPL 4.

Please note that most Type 1 fonts are covered by copyright. Owning the right to use a font does not automatically give you the right to use the font with PrintToPDF, because this will embed the font in the PDF file that is created (PrintToPDF does not support font-subsetting). Some vendors impose restrictions on the use of fonts for electronic viewing. It is the responsibility of the Maconomy user to ensure that the use of a given font is permitted.

Font Administration and Installation

A limited number of license-free fonts are available by default. The available fonts are:

- Helvetica
- Times Roman
- Courier
- Symbol

- ZapfDingbats

If you need any fonts other than this standard set, they must be installed on the server. To install a Type 1 (PostScript®) font on the server, perform the following procedures.

To prepare to install a Type 1 font on the server, complete the following steps:

1. Purchase the font, for example, Garamond, and install it on the server.

   As a result of the installation, the files garamond.pfm and garamond.pfb are created.
2. Make Maconomy aware of the .pfm and .pfb files.
3. Assign Maconomy font names to the Type 1 font.

   The actual procedure for font installation varies, depending on the operating system of the Maconomy server. The following steps apply to a Maconomy server that is running supported versions of Microsoft Windows.

To install a Type 1 font on the server, complete the following steps:

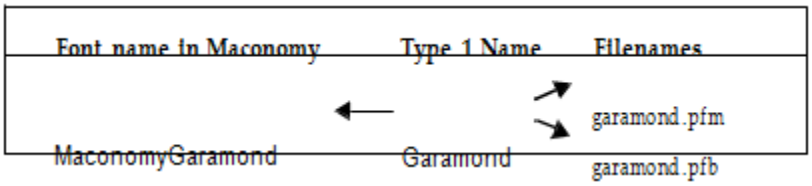
1. Log in as Administrator.
2. Click the **Fonts** icon in the Control Panel.
3. In the Fonts window, select **Install New Font** in the File menu and choose the drive and folder in which your new font was placed.
4. Click **Select All**.
5. Make sure that the **Copy fonts to Fonts folder** check box is selected.
6. Click **OK**.
7. In the progress dialog, click **Yes** for all messages.

For Unix servers, please refer to the documentation of the version of Unix that you are running.

For Maconomy to use the fonts successfully, all that you need are the *.pfm and *.pfb files. The .pfm files contain a description of the metrics—that is, size measurements—of the current font. The .pfb files contain a bitmap rendering of the current font.

Make Maconomy Aware of the Font

To make a font known to Maconomy, first place the .pfm and .pfb files in the Maconomy font directory and then create a mapping from the *Type 1 font name* to the *Maconomy file names*. Then create another mapping from *Maconomy font names* to *Type 1 font names*, as illustrated in the following figure, using the Garamond font as an example.



The Default Font Directory

Maconomy by default looks for fonts in this subfolder:

```
FontSupport/fonts/
```

in the Maconomy server folder, typically Maconomy. Maconomy can be set up to look elsewhere for fonts; please see “Creating Your Own Font Folder.”

To make *fontname* available to the Maconomy server, place the font description files (usually called *fontname*.pfm and *fontname*.pfb) in this folder.

The Font Map File

For Maconomy to know which files to open for the font descriptions, you must create a mapping from the Type 1 font name *fontname* to the font description file names, typically *fontname*.pfm and *fontname*.pfb. This is done in the file fontmap.lst, which can be found in the folder FontSupport located in the Maconomy server folder.

For MPL 4, this is done in the file FontFileMap.lst, which can also be found in the folder FontSupport located in the Maconomy server folder; see "Font Administration and Installation in MPL 4."

You must create an entry at the end of the file containing the font name and the two filenames. The entry consists of one line, where the names are separated by <tab>, and the filenames are enclosed by sets of parentheses. A comment line can be inserted by prefixing it with "#," for example:

```
# Font Name   PFM File      PFB File
Garamond      (Garamond.pfm) (Garamond.pfb)
```

This entry specifies that the font with Type 1 name *Garamond* is defined using the files Garamond.pfm and Garamond.pfb, both located in the folder FontSupport/fonts/ or in a custom font folder.

Create Your Own Font Folder

By default, the location of font files is relative to the folder FontSupport/fonts, but you can specify another location for font files in the file fontmap.lst (in the folder FontSupport in the Maconomy server folder).

Two lines must exist in the file fontmap.lst if you want to specify a different font folder. The first line specifies prefixes in the font location. If the font location contains any of these prefixes, the font location will be regarded as absolute, that is, not relative to FontSupport/fonts.

The second line specifies the actual location in which Maconomy should look for pfm and pfb files, respectively.

Example

```
$Abs_Path_Prefix     (\)     (.)     (C:)    (D:)
$Font_Directories (C:\psfonts\pfm)       (C:\psfonts\pfb)
```

This tells Maconomy to look for fonts in C:\psfonts\pfm and C:\psfonts\pfb. The specification C:\psfonts\fonts begins with C:, which was specified in the $Abs_Path_Prefix line, and is therefore considered an absolute path.

The location of .pfm and .pfb files need not be the same. However , for MPL 4, the location of the files must be the same; please see "Font Administration and Installation in MPL 4."

The items are listed on the same line, separated by <tab>. You can specify a maximum of ten items on a line.

Assume that you want to place font descriptions in a folder called C:\pdffonts\, and you want the .pfm files in a subfolder called pfm and the .pfb files in another subfolder called pfb. Then the path specification section of the file should look like the following:

```
$Abs_Path_Prefix     (\)     (.)     (C:)    (D:)
$Font_Directories (C:\pdffonts\pfm)      (C:\pdffonts\pfb)
```

Note that the folder specification must not be followed by a folder separator character, such as \ or /.

These two lines that specify font folders and absolute prefixes must be specified before the font map specifications.

Using the preceding definition, you can specify a font map (see “The Font Map File”) like this:

```
Marlett       (marlett.pfm) (marlett.pfb)
Garamond      (C:\test\Garamond.pfm)(C:\test\Garamond.pfb)
```

Maconomy first checks whether the Marlett font is a built-in font. Then it checks whether Marlett is in the default FontSupport\fonts folder in the Maconomy server installation, and finally in the path specified by the $Font_Directories directive.

For Garamond, Maconomy performs the same three checks as for the Marlett font. If the font is not found, it searches the specified path (C:\test). The fact that the filenames start with C:\ (which is specified in the $Abs_Path_Prefix line) indicates that these are filenames with an absolute path.

Specify Maconomy Font Names for Type 1 Fonts

Performing the steps described in “Making Maconomy Aware of the Font” makes fonts known to the Maconomy system by their Type 1 font names. However, when you create or compile MPL print layouts, names other than the Type 1 font names are used for the fonts in the MPL specification.

As an example, installing the Type 1 font *Times* actually installs four fonts with distinct Type 1 font names: *Times-Roman*, *Times-Bold*, *Times-Italic*, and *Times-BoldItalic*. The MPL specification uses a different concept. Here, there is only one font called Times, to which attributes such as bold and italic can then be applied (underline is handled differently, and is not part of the font). Furthermore, you might want to be able to refer to this font both as *Times* and as *Times New Roman*.

To create this mapping between Maconomy font names and Type 1 font names, you need to edit the file FontAttributeMap.lst, which is found in the FontSupport folder in the Maconomy server folder. The entries for *Times-Roman* as discussed previously are the following:

```
Times-Roman   Times
Times-Roman   Times New Roman
Times-Bold    Times, BOLD
Times-Bold    Times New Roman, BOLD
Times-Italic  Times, ITALIC
Times-Italic  Times New Roman, ITALIC Times-BoldItalic       Times, BOLDITALIC
Times-BoldItalic    Times New Roman, BOLDITALIC
```

Each line contains two texts separated by <tab>. The first text is the Type 1 font name, and the second is the Maconomy font name. The Maconomy font name can be suffixed with:

```
, BOLD
, ITALIC
, BOLDITALIC
```

The suffixes must be written exactly in this way, including the space after the comma. No other font-modifying attributes can be specified.

The first line maps the Maconomy font name Times (without any attributes) to the Type 1 font name *Times-Roman*. The second line in effect creates an alias to the MPL Times font by specifying that the Type 1 font *Times* is also called Times New Roman in MPL. The third line

maps the same Maconomy font name (that is, Times), but with the bold+ attribute set, thus mapping the font + attribute combination to the Type 1 font *Times-Bold*.

The Times font is already defined in the FontAttributeMap.lst file, but similar lines must be added when new fonts are installed on the system.

Please note that neither the MPL 2 nor the MDL compiler checks for the existence of fonts. Hence, the following is a valid statement in MDL and MPL 2:

```
"Hello":fontname="thisisnofont"
```

Therefore it is important to be careful when specifying font names, as typos such as "Hello":fontname="Helvetica" or similar will go unnoticed by the compiler. See “Fonts in MPL 2 and MDL.”

PrintToPDF Initialization File

Some of the Maconomy PDF and MPL 2 compiler functionality can be modified by changing the parameters of the file PrintToPDF.i, which can be found in the IniFiles folder in the Maconomy folder. Please note that the MPL 2 compiler also makes use of this file, but the MPL 4 compiler does not.

The valid parameters are summarized in the following tables. Parameters are set by stating parametername=value.

	Description	Default Value
FONATTRIBUTEMAPDIR	Path to the folder where FontAttributeMap.lst is found.	FontSupport in Maconomy folder
CPDFFONTMAPDIR	Path to the folder where FontMap.lst is found.	FontSupport in Maconomy folder.
COMPRESSION	Compression (1=on/0=off).	1
LOGLEVEL	Logging (1=on/0=off).	0
MACTOWINDOWSCORRECTION	Do not change this.	18
ISLANDTEXTMODIFICATIONHORIZONTAL	Do not change this.	4
ISLANDTEXTMODIFICATIONVERTICAL	Do not change this.	4
UNDERLININGCORRECTION	Do not change this.	2
SUBSTITUTIONFONT	If a font cannot be found, use this font instead.	Times-Roman
FONTWIDTHSCALFACTOR	Multiply calculated text widths with this factor when formatting MPL layouts.	1

The following is a description of the PrintToPDF.i parameters.

- **FONTATTRIBUTEMAPDIR and CPDFFONTMSAPDIR** — By setting these parameters, you can change the folder path to the location of the FontAttributeMap.lst and fontmap.lst files, respectively. Please note that no path delimiter (/ or \) should be specified at the end of the path name. There is rarely any need to change any of these parameters.
- **COMPRESSION** — This parameter indicates whether the printer driver PrintToPDF should generate compressed (smaller) PDF.
- **LOGLEVEL** — This parameter indicates whether the printer driver PrintToPDF should generate a log file while running. The log file can be used for debugging purposes.
- **MACWINDOWSCORRECTION, ISLANDTEXTMODIFICATIONHORIZONTAL, ISLANDTEXTMODIFICATIONVERTICAL, and UNDERLININGCORRECTION** — These parameters are constants used by PrintToPDF to adjust the prints. This adjustment is necessary due to deviations between Macintosh and Windows formats. Do not change these; they are carefully balanced.
- **SUBSTITUTIONFONT** — This parameter specifies the Type 1 name of the font that should be used if the specified font cannot be found, both when printing using PrintToPDF and when calculating text widths in the MPL 2 compiler. The MPL 4 compiler does not use the notion of substitution fonts.
- **FONTWIDTHSCALEFACTOR** — This parameter is only used by the MPL 2 compiler. When calculating the width of a text in a given font, the result is multiplied by this factor. This parameter is a work-around to a problem concerning Windows and Macintosh fonts: Windows (True Type) and Type 1 fonts are very similar, whereas Macintosh fonts are often wider. Since the calculation of the width of texts is performed using Type 1 fonts, the consequence is that texts may be cut off when printing from a Macintosh computer. A FONTWIDTHSCALEFACTOR of approximately 1.2 (perhaps less, depending on the fonts used) seems to remedy this problem. Note, however, that this creates extra spacing when printing from Windows, hence this workaround should only be used when necessary (that is, when there is a need to print from a Macintosh computer).

PDF Printing

The PDF printer driver is called PrintToPDF.exe and is located in the bin folder of the Maconomy server folder on Windows. The following information is valid as of version 1.19 of PrintToPDF.exe.

The syntax for calling the PDF printer driver is as follows:

```
PrintToPDF.exe [-d] [-c <filename>+] [-l] [-p<papersize>] [- n<filename>+] [-s]
[-h] [-v]
```

Note that there should be no space between flag and argument (for example, between -n and the filename). The parameters have the following meanings:

- **-n** — Input filename. The name of the output file is generated by Maconomy. If the print file is called *name*, the output file will be called *name.pdf* and will be placed in the same folder as the input file *name*. This parameter is mandatory.
- **-c** — PrintToPDF can concatenate multiple print files into one output file, which is written to stdout. In this way, a list of print files can be changed to one print file. For example, assuming that you want to create one PDF file from the following two print files PRINT00001 and PRINT00002, you can enter one of the following:

  ```
  Printtopdf –c PRINT00001 PRINT00002 | printtopdf –s -nFULLPRINT
  ```

  In this way, the concatenated file is piped back to PrintToPDF, which then generates the file FULLPRINT.PDF, which contains both print files. No temporary file is created.

  ```
  Printtopdf –c PRINT00001 PRINT00002 > FULLPRINT
  Printtopdf –nFULLPRINT
  ```

Using this method, the two files are first concatenated into a temporary file. FULLPRINT. PrintToPDF is then invoked again to create the FULLPRINT.PDF file.

Please note that instead of creating a PDF file, the concatenated print file can be sent to another printer driver, for example, the hpgl printer driver.

- **-s** — Read input from stdin. If this parameter is specified, the –n parameter is only used to find the output filename (but –n must be specified).
- **-d** — Debug mode. Print debug messages while running, only used for debugging purposes.
- **-l** — Landscape. Generate the PDF file in landscape mode.
- **-p** — Paper size. Specified as an HP PCL identification for paper size:
  - 1=Executive
  - 2=Letter
  - 3=Legal
  - 26=A4
  - 27=A3
- **-v** — Version information. This shows the date and number of the current version of PrintToPDF.
- **-h** — Help. This shows a summary of available features in PrintToPDF.

A call to PrintToPDF typically only consists of the -n and -p options, for example:

```
PrintToPDF.exe -p26 -nMyPrint
```

This call would generate an A4-sized document in the file MyPrint.pdf, ready to open with Acrobat Reader. Please note that PrintToPDF embeds all fonts in the PDF file.

Fonts in MPL 2 and MDL

There are a number of considerations to take into account when working with fonts other than the standard set (also known as Base 14 fonts)[2]. The following notes are a description of things that you should be aware of when working with different forms of Maconomy layouts.

MPL

The MPL 2 compiler uses the font description files of the Type 1 fonts installed on the server for calculating the width of texts on a print layout. This information is then used during formatting to ensure that there is room for all texts, while not wasting space.

When the MPL 2 layout is printed from the client, the fonts installed on the client computer are used. If a font referenced in the layout does not exist on the client, the substitution font is used. This may cause texts to be cut off, if the font metrics of the substitution font are greater than the font on the server which was originally referenced.

If the font specified in the MPL 2 layout does not exist on the server when the layout is compiled, the font metrics of the substitution font are used. When this happens, no error message is given. The reason that MPL 2 allows you to create layouts with reference to non-existing fonts is that you may not want to add the font to the server for reasons of the cost or work involved in installing the font. If you know in advance that you are referencing a font that does not exist on

[2] The standard Type 1 fonts known to the system are *Helvetica*, *Times-Roman*, *Courier*, *Symbol* and *ZapfDingbats*.

the server, you should manually supply sufficient room for the text in your layout, using the width attribute, for example, like this:

```
"Barcode text":fontname="OCR-B":width=10cm
```

An example of this may be a layout for use in the warehouse, which contains a reference to a bar code font such as OCR-B. The layout is only used on one machine, where the OCR-B font is installed, and will thus print correctly from that computer only.

However, this can be a source of error in the layouts. If, for instance, you meant to reference a font called Marlett that is properly installed on the server, but you inadvertently typed “Martlett,” the MPL 2 compiler will use the substitution font for calculating the text widths, thus not using the correct Marlett font metrics.

Windows clients contain a built-in substitution of the following two fonts:

```
Font:  Substituted By:
Helvetica     Arial
New Century Schlbk  NewCenturySchlbk
```

However, these substitutions can be changed, and more can be added, by changing the file maconomy.ini in the Maconomy folder. By adding a section called FontSubstitutes to the file, you can control the font of the print layouts by setting “Original Font Name”=“Substitute Font Name,” for example:

```
...
[FontSubstitutes] Helvetica=Helvetica BarCodeFont=OCR-B
...
```

The preceding settings specify that if the MPL layout specifies Helvetica, *Helvetica* should in fact be used. If the MPL layout specifies that a barcode font should be used (the name not being known at the time of writing the layout), this particular Windows client should substitute this non-existing font with the proper *OCR-B* font.

On Macintosh clients, the fonts referenced in the MPL layout are substituted with Macintosh fonts using the substitution table embedded in the Macintosh client. Since Macintosh fonts are often slightly wider than the corresponding Windows fonts, the setting `FONTWIDTHSCALEFACTOR` in the PrintToPDF.i file can be used to force the MPL compiler to add extra space to the length of the texts in the layout, thus ensuring that texts are not cut off when printed from a Macintosh computer. A factor of 1.2 is often suitable. However, this setting should only be used if there are in fact Macintosh computers printing from Maconomy.

To change the font substitution in the Macintosh client, you need to edit the dialog handler string list resource #7001 using the ResEdit utility. This resource contains strings in sets of four:

1. Internal font name — The name from the MPL layout, for example, Laudatio.
2. External font name — The name of the Macintosh font that substitutes the internal font name, for example, Geneva.
3. Displacement — Enter 0 for off or 1 for on.
4. Use fractional widths — Enter 0 for off or 1 for on for more accurate spacing.

In the preceding example, whenever the Laudatio font is referenced in an MPL layout, it will be substituted by the Macintosh Geneva font.

When printing using the Maconomy client for the Java platform, the settings in the file PrintToPDF.I are used for generating the PDF file for an MPL 2 printout. MPL 4 does not make use of the PrintToPDF.I file and does not use the notion of substitution fonts.

MDL

The problem of truncated text does not apply to MDL, because dialog layouts are not formatted until the dialog is displayed. As opposed to MPL, which uses text widths that are calculated using the fonts that are installed on the server when the layout was compiled, MDL uses the fonts that are installed on the client machine when calculating the width of texts.

On Macintosh clients, the fonts referenced in the MDL layout are substituted with the Macintosh system fonts as set up on the client computer.

In the Maconomy client for the Java platform, all fonts are displayed using Arial.

Example of Font Installation

Example 1 of Font Installation

Assume that you want to install the font *Poppl-Laudatio*. The font is defined in two files called Pyrg____.pfb and Pyrg____.pfm. First place these two files in the fonts folder (on a Windows server, this is typically C:\MaconomyNT\FontSupport\fonts).

Second, create a mapping between the font name and the font files by creating an entry in the fontmap.lst file (typically C:\MaconomyNT\FontSupport\fontmap.lston Windows). Enter this line:

```
PopplLaudatio (Pyrg__.pfm) (Pyrg__.pfb)
```

The third step is to create a mapping from the Maconomy font name to the Type 1 font name. This entry is created in the file FontAttributeMap.lst file (typically C:\MaconomyNT\FontSupport\FontAttributeMap.lst on Windows). Enter this line:

```
PopplLaudatio PopplLaudatio
```

This concludes the installation of the font. Note the following:

- With the preceding installation, a boldface version of Poppl-Lautdatio is not known to the MPL 2 compiler. This implies that the following:

  ```
  "text":fontname="PopplLaudatio":bold+
  ```

  will be written as Times, BOLD (as this is the default substitution font). We do not have a bold version, but we might want to use the non-bold version of Poppl-Laudatio, even when bold+ is specified. In that case, you should add the following lines to the FontAttributeMap.lst file:

  ```
  PopplLaudatioPopplLaudatio, BOLD PopplLaudatioPopplLaudatio, ITALIC
  PopplLaudatioPopplLaudatio, BOLDITALIC
  ```

- We chose to give the font its proper name, but both the Type 1 font name and the Maconomy font name are free to choose. We could, for example, call the font MyHeadlineFont in the FontAttributeMap.lst file.

Example 2 of Font Installation

Assume that you want to install the font *Frutiger*. The font is defined in two files called FTR____.pfb and FTR____.pfm. First place these two files in the fonts folder (on a Windows server, this is typically C:\Maconomy\FontSupport\fonts).

Second, create a mapping between the font name and the font files by creating an entry in the fontmap.lst file (typically C:\Maconomy\FontSupport\fontmap.lst on Windows). Enter this line:

```
Frutiger      (FTR____.pfm) (FTR__.pfb)
```

The third step is to create a mapping from the Maconomy font name to the Type 1 font name. This entry is created in the FontAttributeMap.lst file (typically C:\Maconomy\FontSupport\FontAttributeMap.lst on Windows). Enter this line:

```
Frutiger      Frutiger 55 Roman
```

This concludes the installation of the font. Note the following:

- With the preceding installation, a boldface version of Frutiger is not known to the MPL 2 compiler. This implies that the following:

  ```
  "text":fontname="Frutiger":bold+
  ```

  will be written as Times, BOLD (because Times is the default substitution font). To use bold or any other font style, we need to specify the bold version of the Type 1 font. To do this, add the following lines to the FontAttributeMap.lst file:

  ```
  Frutiger-Bold Frutiger 55 Light, BOLD Frutiger-Italic Frutiger 55 Roman,
  ITALIC Frutiger-Black      Frutiger 55 Roman, BOLD
  ```

  and so on.

- We chose to give the font its proper name, but both the Type 1 font name and the Maconomy font name are free to choose. We could, for example., call the font MyHeadlineFont in the FontAttributeMap.lst file.

Font Administration and Installation in MPL 4

In addition to Type 1 (PostScript) fonts and a limited number of license-free fonts, MPL 4 also allows the use of TTF (TrueType), OTF (OpenType), and AFM (Adobe) fonts.

Making MPL 4 Aware of the Font

In MPL 4, two mappings need to be created: one mapping between the actual font name and the font filename (for fonts other than Base14 fonts) and another mapping between the actual font name and the Maconomy font name.

The Font Search Path

MPL 4 determines the location of the font files using the MACONOMY_FONTHOME environment variable in the MaconomyServer ini (.I) file. If the MACONOMY_FONTHOME environment variable is not explicitly set, the default font file location is assumed to be the FontSupport/Fonts folder.

Several font search paths can be specified separated by a semicolon, for example:

```
MACONOMY_FONTHOME=D:\maconomy\FontSupport\fonts;C:\WINDOWS\Fonts;C:\m yfonts
```

Thus MPL 4 can use font files placed in the three folders mentioned previously. To create a mapping between the actual font name and the font file name please see "The FontFileMap File." For Base 14 fonts, the MACONOMY_FONTHOME environment variable is not used.

The FontFileMap File

For the MPL 4 compiler and runtime to know which files to open for the font descriptions, a mapping from the actual font name to the font file name must be created. This is done in the FontFileMap.lst file, which can be found in the folder FontSupport located in the Maconomy server folder.

An entry that contains the actual font name and the font file name can be created at the end of the file. The entry consists of a line with the actual font name and the font file name separated by <tab>. A comment line can be inserted by prefixing it with "#," for example:

Font Name	Font File Name
HelveticaUni	HelveticaWorld-Regular.ttf
Garamond	Garamond.pfm

The first entry specifies that the TTF font Helvetical Unicode is defined using the file HelveticaWorld-Regular.ttf. Similarly, the second entry specifies that the Type1 font Garamond is defined using the file Garamond.pfm.

For Type1 fonts, MPL 4 (unlike MPL 2) requires that both .pfm and .pfb files be present in the same directory and, therefore, only the .pfm file name needs to be present as part of the entry in the FontFileMap.lst file as seen in the preceding example.

If a font other than a Base 14 font is used in a MPL 4 layout without a corresponding mapping between the actual font name and the font file name in the FontFileMap.lst file or the font file cannot be found for some reason, a runtime error will be thrown by MPL 4. For Base 14 fonts, the FontFileMap.lst file is not used.

Specifying Maconomy Font Names in MPL 4

In MPL 4, the mapping between actual font names and Maconomy font names is created in a manner similar to MPL 2 by using the FontAttributeMap.lst file (found in the FontSupport folder found in the Maconomy server folder); please see "Specifying Maconomy Font Names for Type1 Fonts." If a Maconomy font name is used in a MPL 4 layout without a corresponding mapping between the actual font name and the Maconomy font name in the FontAttributeMap.lst file, the Helvetica font is used by default.

Defining Default Font in MPL4

In MPL4, a default font can be configured in a custom version of the Maconomy.ini file:

```
[MPLConfig]
DefaultFont=HelveticaUni
```

Example of Font Installation in MPL 4

Example 1 of Font Installation in MPL 4

Assume that you want to install the basic version of the Helvetica Unicode True Type font. The font is defined in a file called HelveticalWorld-Regular.ttf.

1. Place the two files in a folder of your choice, for example, C:\Maconomy\FontSupport\Fonts.
2. Specify that MPL 4 should search for the font files in the preceding folder by adding the following entry:

   ```
   MACONOMY_FONTHOME = C:\Maconomy\FontSupport\Fonts
   ```

   MaconomyServer.I file. (Additional folders to search for font files should be separated by a ";".)
3. Create a mapping between the actual font name and the font file name by creating an entry in the FontFileMap.lst file. Enter the following line:

   ```
   HelveticaUni  HelveticaWorld-Regular.ttf
   ```
4. Create a mapping between the actual font name and the Maconomy font name by creating an entry in the FontAttributeMap.lst file. Enter the following line:

```
HelveticaUni  HelveticaUni
```

5. In addition configure ‘HelveticaUni’ as the default font in a custom version of Maconomy.ini like this:

```
[MPLConfig]

DefaultFont=HelveticaUni
```

If the actual bold, italic, or bold-italic versions of the Hevetical Unicode font are available and intended to be used, steps 1, 3 and 4 in the preceding procedure should be repeated with the corresponding font files.

For example, to use bold, italic, and bold-italic fonts, place the corresponding font files in the font folder and then add the following lines to the FontFileMap.lst file:

```
HelveticaUni-Bold   HelveticaWorld-Regular.ttf

HelveticaUni-Italic HelveticaWorld-Regular.ttf

HelveticaUni-BoldItalic    HelveticaWorld-Regular.ttf
```

And add the following lines to the FontAttributeMap.lst file:

```
HelveticaUni-Bold   HelveticaUni, BOLD

HelveticaUni-Italic HelveticaUni, ITALIC

HelveticaUni-BoldItalic    HelveticaUni, BOLDITALIC
```

And in addition configure HelveticaUni as the default font in a custom version of Maconomy.ini like this:

```
[MPLConfig]

DefaultFont=HelveticaUni
```

Example 2 of Font Installation in MPL 4

Assume that you want to install the basic version of the Garmond Type1 font. The font is defined in the file Garmond.pfm and Garmond.pfb.

1. Place the file in a folder of your choice, for example, C:\Windows\Fonts.
2. Specify that MPL 4 should search for the font files in the preceding folder by adding the entry MACONOMY_FONTHOME = C:\Windows\Fonts to the MaconomyServer.I file. (Additional folders to search for font files should be separated by a “;.”)
3. Create a mapping between the actual font name and the font file name by creating an entry in the FontFileMap.lst file. Enter the following line:

```
Garmond       Garmond.pfm
```

4. Create a mapping between the actual font name and the Maconomy font name by creating an entry in the FontAttributeMap.lst file. Enter the following line:

```
Garmond       Garmond
```

This concludes the installation of the basic Garmond font.

If the bold, italic, or bold-italic versions of the Garmond font are available and intended to be used, steps 1, 3, and 4 in the preceding procedure should be repeated with the corresponding font files.

For example, to use bold, italic, and bold-italic fonts, place the corresponding font files in the font folder and then add the following lines to the FontFileMap.lst file:

```
Garmond Bold  GarmondB.pfm
Garmond Italic       GarmondI.pfm
```

```
Garmond Bold Italic GarmondZ.pfm
```

And add the following lines to the FontAttributeMap.lst file:

```
Garmond Bold  Garmond, BOLD
Garmond Italic      Garmond, ITALIC
Garmond Bold Italic Garmond, BOLDITALIC
```

And in addition, configure Georgia as the default font in a custom version of Maconomy.ini like this:

```
[MPLConfig]
DefaultFont=Georgia
```

Troubleshooting

If you experience problems when installing and using a new font, even after following this guide, you can check the following list. This list contains the questions which, in our experience, come up most often, or the steps that are most often overlooked.

The list is in no particular order.

- MPL 2 supports PostScript fonts also known as Type1 fonts but does not support TrueType (.ttf) or other font types. A PostScript font consists of a set of two files: one with the file type .pfm and one with the file type .pfb. MPL 4 supports True- Type (.ttf), OpenType (.otf), and Adobe Font Metric (.afm) fonts in addition to PostScript fonts.
- Do you refer to the fonts correctly in MPL? When referring to fonts in MPL, remember that font names are case sensitive; the font “Arial” is not used if it is referred to as “arial” in an MPL layout.
- Are you using the correct font specification files? MPL 2 uses the fontmap.lst file for mapping the actual font name to the .pfm and .pfb font file names and the FontAttributeMap.lst file for mapping the actual font name to the Maconomy font name. MPL 4 uses the FontFileMap.lst file for mapping the actual font name to the font file name and the FontAttributeMap.lst for mapping the actual font name to the Maconomy font name. These three files mentioned previously can be found in the FontSupport folder of the Maconomy server folder.
- Are you using spaces or tabs in the font specification files? You must use tabs to separate the information columns in the fontmap.lst and FontAttributeMap.lst files for MPL 2 and in the FontFileMap.lst and FontAttributeMap.lst files for MPL 4.
- Are you expecting a boldface or italic font, but getting the plain font? If you specify, for example, bold+ in an MPL layout, the font must support a boldface variant. This variant must be referenced in the FontAttributeMap.lst file, listing the PostScript variant on the left side and the MPL reference on the right.
- When using, for example, PrintToPDF, an error message says that the font is not found. For instance: ClibPDF: Font not found: OCRB-Alternate >> substituting Times-Roman.

  This usually means that PrintToPDF could not find the font that is referenced in the MPL layout being printed. (It could also mean that the font that it found is not a correct font, but that is quite unusual.) By default, PrintToPDF looks for fonts in the current directory first, then in /usr/local/font/pfm and /usr/local/font/pfb (Unix only), and finally in the folders specified for the font in the fontmap.lst file.
- Have you set the MACONOMY_FONTHOME environment variable in the MaconomyServer.I file for MPL 4?

MPL 4 uses the MACONOMY_FONTHOME environment variable to search for font files. Multiple folders separated by a semicolon can be specified. If this environment variable is not set, FontSupport/fonts is assumed to be default location of the font files. This environment variable is not used for Base 14 fonts.

- Do you get the Helvetica font in the pdf instead of the font used in the MPL 4 layout?

  This indicates a missing mapping between the actual font name and the Maconomy font name in the FontAttributeMap.lst file and, in such a case, Helvetica is used as the default font. Note that MPL 4 does not support the notion of substitution fonts used by MPL 2.

- Do you get a runtime error when using a font other than a Base 14 font in MPL 4?

  This may be due to a missing mapping between the actual font name and the font file name in the FontFileMap.lst file, or because the folder in which the font file is placed is not specified in the MACONOMY_FONTHOME environment variable or because the font file could not be found for some reason.

Kona Integration

Maconomy is integrated with Deltek's Kona Business solution through Workspace Client. Kona is a private and secure collection of spaces that allows you to collaborate on your projects with your team members. This cloud-based social collaboration and productivity platform enables users and groups to connect, organize and work together through easy-to-use conversation spaces, task lists, common calendars, and file-sharing.

Kona integrates with Maconomy to provide a seamless means for integrating your Maconomy with your Kona spaces. This allows you to share projects with team members and view related online conversations, project-related tasks, calendar events, and files.

With Kona in place within Maconomy, conversations, key deliverables, important documents, and major milestones are all collected and unified in context so teams can identify issues, solve problems, and keep everyone up to date on the status of deadlines and deliverables. Kona's integration with Deltek Maconomy provides a powerful solution to better support your business workflows and processes and enable more efficient collaboration and productivity on projects.

A Kona Business Edition license is required before the Maconomy installation is started to set up the data that is needed for the Kona integration using MConfig.

Generate Client ID and Client Secret to be used during MConfig Installation

An administrator for the Kona Business account can get the Client ID and Client Secret.

To get the Client ID and Client Secret, complete the following steps:

1. Start Kona.
2. In Kona's left sidebar, click **Accounts**.
3. Click the account name.
4. Click the **Edit Account** button in the upper-ride side of the window.
5. Click the **Integrations** tab.
6. Click **Add application**.
7. In the **Application Name** field, enter **Maconomy**.
8. In the **Redirect URL** field enter **https://www.kona.com/api/token**.

9. Click the **Done** button.
10. Copy the values for Client ID and Client Secret to the relevant fields during the installation of Maconomy using Mconfig.

Set Up Kona Integration in Maconomy

To set up Kona integration, you must create a login within Kona, and then integrate the functionality by enabling a new parameter within Maconomy.

Go to Kona.com for details on using Kona, including creating a login.

To enable Kona integration within Maconomy, complete the following steps:

1. Click **System Setup » Parameters and Numbers.**
2. Select the **Enable Kona Integration** parameter, and click **Save**.

Use Kona Within Maconomy

To use Kona from within Maconomy, complete the following steps:

1. Click **Jobs and Opportunities » Home**.
2. On the **Information** tab, the new Kona group displays.

Upon initial use, a Kona user login box appears within the Maconomy Workspace client. Enter your Kona email address and Kona password to access Kona directly from Maconomy.

3. Use the **Kona Space** drop-down field to select from the list of all Kona spaces available to you. Select the relevant space to display inside the Workspace Client, using the embedded browser.

You must set up your Kona spaces from within Kona.

4. Use Kona as needed from within the Workspace Client, including carrying integration over to new jobs you create, as follows:
   - When creating a job, the Kona space carries over from the template job.
   - When copying a job, the Kona space is copied to the new job.
   - When converting an opportunity to a job, the Kona space is copied to the created job.

Getting Started in Kona

Share a Job with Kona

Kona is a collection of spaces that you can use to share conversations, events, tasks, and files. You can associate a Kona space with a Maconomy job and share the current project's information.

To associate a Kona space with a job in Maconomy, you must also be a member of the Kona space.

To share a job with Kona, complete the following steps:

1. Open a job in Maconomy.
2. Click **Jobs and Opportunities » Home » Information** tab.
3. Open the Kona browser panel in the Workspace Client to view the Kona space.
4. On the Change Kona Space dialog box, use the lookup to select the Kona Space that you want to associate with the project.
5. Click **Finish**.

You can associate only one Kona space with each Maconomy job. When you select the job in Maconomy and open the Kona pane, the associated space displays.

View Conversations in Kona

The Kona Conversations tab displays the conversations (the online discussions) for the current space. You must be configured to use Kona before you can view conversations in Kona.

To view a conversation in Kona, complete the following steps:

1. Click **Jobs and Opportunities » Home » Information** tab.
2. Open the Kona browser panel in the Workspace Client to view the Kona space.
3. Click the Conversations tab.
4. Click a conversation's title to view its details.

View Tasks in Kona

The Kona Tasks tab displays the tasks that are related to the current space. You must be configured to use Kona before you can view tasks in Kona.

To view a task in Kona, complete the following steps:

1. Click **Jobs and Opportunities » Home » Information** tab.
2. Open the Kona browser panel in the Workspace Client to view the Kona space.
3. Click the Tasks tab.
4. Click a task's title to view the task's specific details.

View Files in Kona

The Kona Files tab displays files for the current space. You must be configured to use Kona before you can view files in Kona.

To view a file in Kona, complete the following steps:

1. Click **Jobs and Opportunities » Home » Information** tab.
2. Open the Kona browser panel in the Workspace Client to view the Kona space.
3. Click the Files tab.
4. Click a file's title to open the file.

View Events in Kona

The Kona Events tab displays the calendar events for the current space. You must be configured to use Kona before you can view events in Kona.

To view an event in Kona, complete the following steps:

1. Click **Jobs and Opportunities » Home » Information** tab.
2. Open the Kona browser panel in the Workspace Client to view the Kona space.
3. Click the Events tab.
4. On the list of calendar events for the current month, click an event's title to view the event's details.
5. Use the arrow buttons to scroll through different months on the calendar.

Change Kona Space Dialog Box

Use the Change Kona Space dialog box to associate a Kona space with a job.

To display the Change Kona Space dialog box, complete the following steps:

1. Click **Jobs and Opportunities » Home » Information** tab.
2. Open the Kona browser panel in the Workspace Client to view the Kona space.
3. Use the Kona Space lookup to select the Kona space that you want to associate with the job.

To remove the association between a job and a Kona space, clear the Kona space field and click **Enter**.

People Planner

People Planner is a separate Deltek offering that can be used with Maconomy.

People Planner bridges the gap between project planning and resourcing planning within professional services organizations.

General Server Maintenance

This information is in the process of being updated for Maconomy Release 2.2. Note that while this update is in progress, not all of the material in this section meets Release 2.2 functionality.

This chapter describes the routines that are necessary for maintaining a Maconomy server, including backing up and restoring a Maconomy installation. The control routines that are described must be performed to check that the Maconomy installation is in good shape. The routines are grouped by the schedule by which they should be performed. Some must be performed daily, others weekly, still others monthly, and finally, some should be performed every six months.

This document is intended for system and database administrators, and to perform these tasks, knowledge at that level is required.

Note that all commands in this document should be performed as the local user `Maconomy`. You should not use `root` (on UNIX) or `Administrator` (on Windows), unless you are completely confident with the commands that you are performing.

In this document it is assumed that the temporary area assigned to Maconomy is `<UserHome>tmp`, for example, `D:\maconomy\Tmp` on Windows. Consult your server installation documentation for the precise assignment of temporary directories.

Much of the information in this chapter pertains to backup and restore. For more information about backup procedures, see “Backup and Restore Guidelines” in this manual.

Database Surveillance

The data of a Maconomy system is placed in a database. The data of different Maconomy companies are placed in separate DB-user accounts in the same database instance. An important part of keeping a Maconomy system functional is to monitor the database.

For information about supported databases, please see “Database and OS Requirements” in this manual. Currently, this document only describes Oracle.

Tablespaces

All objects (tables, indexes, and so on) in an Oracle database are stored in tablespaces. The data in each tablespace is stored physically in one or more data files.

Maconomy uses the following six tablespaces.

Maconomy	Used as the default tablespace. All Maconomy data and indexes are stored in this tablespace.
MacTmp	Used as the temporary tablespace. A temporary tablespace holds temporary segments, which are created automatically and used during SQL execution.
MacRollback	Holds the rollback segments used during SQL execution.
MacIndex	Holds the indexes for all the relations/tables of the database.
System	Holds system information for the instance and is not described further in this document.
Oem_Repository	Holds system information for the instance and is not described further in this document.

When the database that Maconomy uses is first created, each of these tablespaces consists of one file. These files must never, under any circumstances, be deleted.

As more data is entered in the Maconomy system, a tablespace can become full. When this happens, it must be extended; otherwise Maconomy will stop functioning. For information about doing this, please refer to “Extending an Oracle Tablespace.”

Data Files

The data in the Oracle database is stored in a number of files that are usually placed in a directory named `Oradata` inside the Oracle Home. Oracle Home is usually a directory named

`Oracle` at the root of a drive. A typical example of the path to the directory where the data files are placed is:

Win NT/Win2K: `C:\Oracle\Oradata\ORCL8`

Unix: `/data/oradata/orcl8`

If more than one physical disk exists in the system, the data files may be placed elsewhere. Use CheckOracle (see the following information) to locate the data files.

CheckOracle

Use the `CheckOracle` utility to check the condition of the Maconomy database.

To run `CheckOracle`, complete the following steps:

1. Open a command prompt/shell.
2. Run the command `CheckOracle` system/manager.

   If you are running Windows, and the command is not recognized, change to the <MaconomyHome>\bin directory before running the command. The output of the command should be similar to the following:

```
D:\maconomy\bin>checkoracle system/manager
CheckOracle version 4.2. Copyright PPU Maconomy 1997-2004.
-- TABLESPACE TOTAL  FREE   LARGEST      USED   % FREE RM
-- -------------    ---------- ---------- ---------- ---------- ------
-----------
-- SYSTEM       327155712      573440     573440    326582272
.175280449 AX D
-- MACROLLBACK  463470592     1048576    1048576    462422016
.226244344 AX D
-- MACONOMY    4196401152   640811008  560922624   3555590144
15.2704898 AX D
-- MACTMP                                                  0
-- MACINDEX    6287261696  1903820800  889978880   4383440896
30.2806037 AX D
---- ----
-
Checking init parameters:
-
..
.. !!! WARNING: DATABASE STATISTICS IS POOR. PLEASE IGNORE TUNING
COMMENTS
Done.
```

This shows information about the various tablespaces in the database:

- `TABLESPACE`

  The name of the tablespace.

- `TOTAL`

  The total allocated size of the current tablespace, i.e. the sum of the columns `FREE` and `USED`, in bytes.

- `FREE`

The size of the available portion of the current tablespace, in bytes.

- `LARGEST`

  The size of the largest object in the current tablespace, in bytes.

- `USED`

  The size of the used (unavailable) portion of the current tablespace, in bytes.

- `% FREE`

  The amount of free space in the current tablespace, as a percentage.

- `RM`

  Any remark to the current tablespace. ”.

Use the information to determine whether one or more of the tablespaces is running full, and you have to add an additional data file to the tablespace in question. When the upper file size limit (2000 MB) is reached, a new file has to be manually added to the tablespace. Please see “Extending an Oracle Tablespace” for information about extending the tablespace.

More Details

If you run `CheckOracle` with the `-a` option: `CheckOracle system/manager -a`, more details are added to the report. Two new sections, `Retrieving details` and `Data files in database`, are added after the tablespace output.

The information in the `Retrieving details` section can be useful for optimizing the Oracle database.

The information in the `Data files in database` section is used to find out which data, control and redo-log files exist in the database. At the bottom of the new output section you will find information similar to the following (depending on platform):

- Windows

```
.. Data files in database
.. Datafiles
.. C:\ORACLE\ORADATA\ORCL\SYSTEM01.DBF
.. C:\ORACLE\ORADATA\ORCL\MACROLLBACK.DBF
.. C:\ORACLE\ORADATA\ORCL\MACDATA01.DBF
.. C:\ORACLE\ORADATA\ORCL\MACTMP01.DBF
.. C:\ORACLE\ORADATA\ORCL\MACINDEX01.DBF
.. C:\ORACLE\ORADATA\ORCL\OEMREP01.DBF
.. Controlfiles
.. C:\ORACLE\ORADATA\ORCL\CONTROL01.CTL
.. C:\ORACLE\ORADATA\ORCL\CONTROL02.CTL
.. C:\ORACLE\ORADATA\ORCL\CONTROL03.CTL
.. Logfiles
.. C:\ORACLE\ORADATA\ORCL\REDO1_1.LOG
.. C:\ORACLE\ORADATA\ORCL\REDO1_2.LOG
.. C:\ORACLE\ORADATA\ORCL\REDO1_3.LOG
..
```

- UNIX

```
.. Data files in database
.. Datafiles
```

```
.. /data/oradata/orcl8/system01.dbf
.. /data/oradata/orcl8/MacRollback.dbf
.. /data/oradata/orcl8/MacData01.dbf
.. /data/oradata/orcl8/MacTmp01.dbf
.. Datafiles
.. /data/oradata/orcl8/MacIndex01.dbf
.. /data/oradata/orcl8/oemrep01.dbf
.. Controlfiles
.. /data/oradata/orcl8/Control01.ctl
.. /data/oradata/orcl8/Control02.ctl
.. /data/oradata/orcl8/Control03.ctl
.. Logfiles
.. /data/oradata/orcl8/redo1_1.log
.. /data/oradata/orcl8/redo1_2.log
..
```

This list shows the following files: one data file for the system tablespace, one data file for the Maconomy temporary tablespace, two data files for the Maconomy rollback tablespace and one data file for the Maconomy data tablespace. Additionally, the locations of the Oracle Control file and the Oracle Redo-log files are shown.

Please note that all these files must be included in an Oracle file system backup.

Autoextend Option

Since version 8, Oracle has had the option to auto-extend the tablespaces as they are running full. If auto-extend is turned on, the only valuable information from `CheckOracle` is to see if an additional data file has to be added to the tablespace in question.

However, the Oracle autoextend option in its current implementation has some drawbacks. First of all, Oracle auto-extends the tablespace in increments of 1 MB. This is a very small amount, especially if we are dealing with large databases, perhaps more than 10 GB, where you would want to increase the tablespace by for example 2 GB at a time. The consequence of these increments being so small is that a large number of data files a created, and that the amount in the `% FREE` column often shows a very small amount, often around 0.5%.

Advocates for the autoextend option say that the option reduces the need for data- base service. But in the end it is up to you, the database administrator, to choose whether you want to turn the option on or not.

Extending an Oracle Tablespace

If a tablespace is running full, you need to extend it. On Windows, this is done using the Oracle Database Administrator program (`svrmgrl`).

In the examples below, we extend the tablespace named `Maconomy`. We assume that the data files are placed in the directory `C:\Oracle\Oradata\ORCL8`, and the data file to add is named `MacData02.dbf`. Do the following:

1. Open a command prompt/shell.
2. Launch `svrmgrl`.
3. Type the following commands:
   - Windows:

     ```
     connect system/manager alter tablespace Maconomy
     ```

```
add datafile 'C:\Oracle\Oradata\ORCL8\MacData02.dbf' size 100 M;
exit
```

- UNIX (only the path notation is different):

```
connect system/manager alter tablespace Maconomy
add datafile '/data/oradata/orcl8/MacData02.dbf' size 100 M;
exit
```

This will add a new data file with a size of 100 MB to the tablespace `Maconomy`.

As Oracle may be initially set up to auto-extend the data files when needed, we only have to observe that the used space in tablespace `maconomy` is lower than 4000 MB. When the used space is close to 4000 MB, an additional file has to be added, and so on.

Please note: If you add a data file to a tablespace, it cannot be removed again without deleting the entire tablespace. Furthermore, it is difficult to rename or move a data file. Consequently, you *must be sure* that you add the file to the correct tablespace, that the file is placed in the correct location, and that the file is given the correct name. Also, check for sufficient disk space before adding a datafile!

You should follow these naming conventions when adding data files to tablespaces:

- Files added to tablespace `Maconomy` are named `MacData<SeqNo>.dbf`.
- Files added to tablespace `MacTmp` are named `MacTmp<SeqNo>.dbf`.
- Files added to tablespace `MacRollback` are named `MacRoll<SeqNo>.dbf`.
- Files added to tablespace `MacIndex` are named `MacIndex<SeqNo>.dbf`.
- Files added to tablespace `System` are named `System<SeqNo>.dbf`.

In all five cases, `<SeqNo>` is a sequence number starting from 2, and using two digits. For example, when adding a data file to the `Maconomy` tablespace for the first time, the file should be named `MacData02.dbf`.

Prior to adding data files, you should use `CheckOracle` with the `-a` parameter to obtain a list of the names of the existing data files. Please see “Check Oracle” for information about using `CheckOracle`.

Operating System Surveillance

To keep the Maconomy server in shape, you must carefully monitor your operating system. For information about supported operating system versions, please see “Database and OS Requirements” in this manual.

What to Watch

Check disk space regularly (see “Check for Free Disk Space”). If you are running out of disk space, check the Maconomy `Tmp` directory for files which can be cleared. For more information, see “Check the Temporary Area for Old Files.”

Regular Routines — Daily

In this section, you will find the list of checks that should be performed regularly. Performing these checks according to the schedule cannot completely ensure the absence of problems, but following the guidelines below will allow you to catch most issues before they become real problems.

Check the Backup

A check of the backup should be performed every day, whether it is performed by Maconomy's backup script or by a third-party tool.

Check the Maconomy backup log to ensure that the backup has run properly. Errors should be reported to the Maconomy Customer Support Service.

The backup log is stored in:

- Unix

  `/usr/Maconomy/backup`

  To check for errors look in the bottom of the `BackupGeneral.log`. If there are any errors, check the `Backup.<date>.log`.

- Windows

  See the installation documentation for the backup log location.

  Check the file `BackupLog.txt` for errors; the latest entry is at the bottom of the file.

Also remember to change the backup tapes.

Check that the Daemons are Running

As described in the “System Overview” chapter of this manual, Maconomy uses daemons to handle requests from the clients and clean up closed connections. The daemons are running on the Maconomy server.

To check the daemon status:

- Unix

  Use `ps –ef | grep Daemon`.

  This will show a list of all processes that have “Daemon” in their names.

  The daemon also has a temporary file. Check that it has been refreshed recently.

  `ls –l /tmp/zDaemon*`

  Examples:

  ```
  maconomy 6 % ps -ef | grep Daemon
  maconomy 23800 22372  0  27 okt      - 11:28 bin/Daemon.r -s30
  maconomy 17584    1  0  28 okt      - 6:06
  /data02/app/maconomy/bin/WebDaemon.nm.4105
  ```

  This output shows both the Maconomy daemon (`Daemon.r`) and a webdaemon. You can check the number of webdaemons and their port numbers in the server installation documentation.

  ```
  maconomy 9 % ls -l /tmp/zDaemon*
  -rw-r--r--       1 maconomy maconomy 0 02 dec 11:57
  /tmp/zDaemon.tmp
  ```

  The time stamp of the file is important. It should have been modified within approximately the last 15 minutes.

- Windows

  In Windows, the Daemons run as services and can be checked in the server’s Services management console window. The services in question are:

```
MaconomyDaemon WebDaemon.<shortname>.<port>
```

Check the Temporary Area for Old Files

Normally the Maconomy application will clean up its old files from the temporary area assigned to it. This is however not the case if the application or the batch job terminates unexpectedly.

The file `DaemonConfig` in the `IniFiles` directory specifies which files will be deleted automatically by the Maconomy daemon. The file contains a number of lines of the following format:

```
<Folder>\<Wildcard> <No.Hours>
```

For instance, the file can contain the following lines (Windows):

```
%MaconomyTmpDir%\PRINT* 6
%MaconomyTmpDir%\ERROR* 24
%MaconomyTmpDir%\MSL* 6
```

According to that setup, files in the Maconomy temporary directory which begin with `PRINT` or `ERROR`, respectively, will be deleted by the `MaconomyDaemon` after 6 and 24 hours, respectively. For UNIX, only the folder separator is different—exchange the

```
\ with a /.
```

Files of the type defined below, which are older than 24 hours, can and should be removed. If you are in doubt, you can ask Maconomy A/S if you can delete a file.

You may find the following file types in the temporary area:

1. Debug files from the Maconomy Daemon (for example, `DdbugDOCAFhIAAA`). Only Maconomy should set the daemon up to run with debugging and we will only do it if necessary. Therefore you should not expect to see files of this type.
2. Error files from the clients (for example, `ERRORZIAFhIAAC`). These files are created for each client and for each job. They will be deleted when a job has finished, unless it terminated abnormally. The contents of these files will correspond with an entry in the `PPU_DebugStr` file.
3. Batch job files (for example `JOBBIKAFhIAAB` or `JOBBIKAFhIAAB.cmd`). These are executable files created for each batch job. Unless the job is terminated abnormally, the files will be removed automatically, otherwise they can be used for localizing the error that caused the termination of the job.
4. Debug files from the Maconomy server process belonging to a batch job, for example, a print file (for example, `JdbugBIKAFhIAAB`). Unless there is a reason for Maconomy to be running with debug switched on, there should not be any files of this type.
5. Debug files belonging to a client, for example, a print file (for example, `OdbugZIAFhIAAf`). Unless there is a reason for Maconomy to be running with debug switched on, there should not be any files of this type.
6. ASCII representations of a printout (for example, `PRINTZIAFhIKAf`). These files are either sent to the client or printed directly from the server.
7. Batch job parameter files (for example, `PRMBIKAFhIAAB`). These files contain the parameters for the batch jobs and will be deleted when the job has finished successfully. The suffix will match a `JOB*` file.
8. Debug files from the Maconomy server process belonging to a report (for example, `RdbugZIAFhIKAf`). Unless there is a reason for Maconomy to be running with debug switched on, there should not be any files of this type.

9. Files created in connection with installation or upgrade of Maconomy. For example:

   ```
   ConvertAnalyzerFiles.<AppName>.<shortname>.txt,
   CreateAnalyzerViews.<shortname>.txt, and LayError_<shortname>.txt.
   ```

   Files of these types concern the installation, upgrade, and validation of Analyzer files and print layouts.

10. Other files may exist, depending on the server and which server features are used.

    If you notice such files, there may be a configuration error in your system, which you should investigate. At any rate, the files can also be deleted if they are, for example, a few days old.

    To check the temporary area:

    - Unix

      ```
      cd /tmp ls –l
      ```

      - To order it by date and time:

        ```
        ls -l -t | more
        ```

      - To remove the files use:

        ```
        rm <filename>
        ```

        DO NOT use wildcards such as `*` or `?` in the filename when deleting.

      - In some cases, the `/tmp` directory can get so full that you are not able to do a directory listing, and hence not able to view and delete the files. In such cases, you can use commands such as the following (the below syntax is AIX):

        ```
        find /tmp -name 'PRINT*' -mtime +1 -exec rm {} \;
        ```

        The above command deletes all printouts (which start with `PRINT`) which are more than two days old. You can repeat this command using for example `PRM` instead, until you can do a normal directory listing.

    - Windows

      Use Explorer and order the files by clicking on the “Modified” tab for the folder.

Check the Maconomy Error Log

Every time something unexpected happens with the Maconomy application, an entry is appended to the log file. The log file will be created if it does not yet exist. On Unix, it is called `PPU_DebugStr`, and on Windows it is `PPU_DebugStr.txt`. It is located in Maconomy’s temporary area.

It may be a good idea to rename the file when it reaches about 1 MB in size. A good naming method is to add the date to the end of the file (but before any extension).

The file is an ordinary text file and can be opened with any text editor. The following is an example of the error log:

```
-----------------------------------------------------
PPU_DebugStr called at Wed Nov 19 09:54:11 2003

Message:
DoReceive: recv failed; errno=73 ( connection to client lost)

Program name:  macoracle.r.w_8_0 Version string:   32.02.0.49
```

```
Process Id:    21128
User name:     Administrator Address, local: 172.16.2.15:4484
Address, remote: 172.16.100.117:1090 Client type:  Windows
Ini file prefix: macoracle.r.w_8_0
```

The entry shows the date, user name, the address of the client, and an error message that describes the problem.

Error Types

Here are some of the most common error types. Note that information about program name, version, address, and so on, has been removed from the examples.

```
-------------------------------------------------------
PPU_DebugStr called at Wed Feb 27 15:02:03 2002

Message:
Debug message received from client: Maconomy client version: */T US
3.2.2-32

AllbaseCom Received ErrorCode 150010
```

This error means that a file could not be found. The error is most often seen on Unix after a failure during backup where the folder `FontSupport` is temporarily renamed.

```
-------------------------------------------------------
PPU_DebugStr called at Thu Nov 21 13:52:51 2002

Message:
ORA-01034: ORACLE not available
```

This error shows that Oracle is not running or that the client is unable to find Oracle. Besides from Oracle being down, this error can be caused by network problems or by the client connecting with the wrong data.

```
-------------------------------------------------------
PPU_DebugStr called at Thu Nov 21 13:52:51 2002

Message:
ORA-1652: unable to extend temp segment by 19762 in tablespace
MACINDEX
```

If an error is caused by the database, the error message will be transferred to `PPU_DebugStr` with a description of the error. In this specific case, the area in the database reserved for indices has run full and needs to be extended. This is a job for a Database administrator. Maconomy Corp. is able to help extending the tablespace. This error can also happen for other tablespaces.

```
-------------------------------------------------------
PPU_DebugStr called at Tue Sep 03 19:46:33 2002

Message:
Server timeout. 240 minutes passed without activity
```

This error happens after 240 minutes of inactivity, where the server closes down the connection. The time-out can be changed; see “Options and Parameters” in “Maconomy Server Options and Parameters.” This error can be ignored.

```
-----------------------------------------------------
PPU_DebugStr called at Thu Sep 05 11:36:53 2002 Message:
ORA-02091: transaction has been rolledback;ORA-00001: unique
constraint (TEST71.UQ0448) has been violated
```

From version 7.0 of the Maconomy application, Maconomy uses constraints in the database to enforce uniqueness on the key fields of the database. In this case, there has been a violation. Unless this error occurs frequently, it can be ignored.

Check for Free Disk Space

The file systems on Unix and the hard disks in Windows should not be allowed to run full. This may affect the execution of processes, for example, printouts and server processes. Especially the drives/file systems where the database’s data files, archive files and the temporary area of Maconomy are placed should be closely monitored.

DO NOT move the data files of the database to get more free space, unless you are a database administrator. If you just move them, it will most likely damage the file in such a way that the latest backup must be reinstalled.

- Unix

  Use the command:

  ```
  df -k
  ```

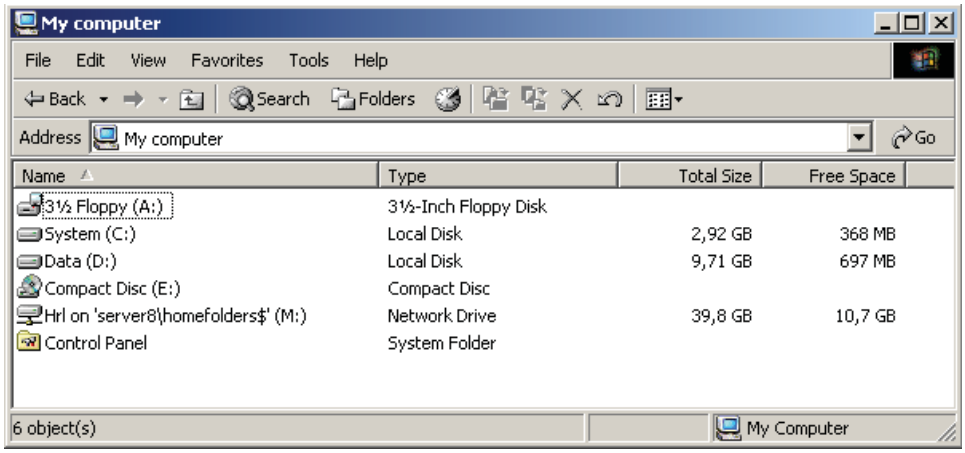
  The result may differ a bit depending on the Unix version, but it will show the results in 1 KB blocks.

Example:

```
maconomy 1 % df -k
Filesystem    1024-blocks   Free     %Used  Iused  %Iused  Mounted on
/dev/hd4      503808        422548     17%  1331       1%  /
/dev/hd2      602112        120660     80%  19130     13%  /usr
/dev/hd9var   102400        75000      27%  232        1%  /var
/dev/hd3      258048        163916     37%  678        2%  /tmp
/dev/hd1      8192          7888        4%  22         2%  /home
/dev/lv00     5013504       2952036    42%  1219       1%  /data01
/dev/lv01     12058624      718028     95%  26234      1%  /data02
```

- Windows

  To check the disk status on Windows, use Explorer. Click “My Computer” and ensure that it shows the results “detailed.”



Regular Routines — Weekly

The routines described in this section should be performed weekly.

Check for Free Space in the Oracle Database

To control that the database is in working order, you can run the `CheckOracle` program. It will print a list of information for each tablespace in the database. If a tablespace is close to full, it should be extended. Usually the following tablespaces need to be extended: `MACONOMY`, `MACINDEX`, and on rare occasions also `SYSTEM`. `MACROLLBACK` will most often look full, but as long as your reports and updates of the system run without problems, there is no need to worry.

`CheckOracle` is a small Maconomy program (used from a command line) and it is stored in the Maconomy `bin` folder.

- Unix

  Example: `/data02/app/maconomy/bin`

- Windows

  Example: `D:\maconomyNT\bin`

The results are similar, except for the path to the data files. To run the program, change your location to the `bin` folder, and type (exchange `manager` for the password for your system user):

```
CheckOracle system/manager
```

The result may look like this:

```
CheckOracle version 4.1. Copyright PPU Maconomy 1997-2002.
-----------------------------------------------------------
Checking tablespace allocation:
-- TABLESPACE  TOTAL FREE  LARGEST     USED  % FREE RM
-- -------------- ---------- ---------- -------- ---------- -----
-
--
-- SYSTEM      122683392   507904      327680      122175488   .4 AX
-- MACROLLBACK 2097152000  180215808   1064960     1916936192  8.6
```

```
**
-- MACONOMY    2298478592  1245978624  4186112     1052499968  54.2
AX
-- MACTMP      524288000   498720768   1064960     25567232    95.1
AX
-- MACINDEX    1918894080  1161723904  1048576     757170176   60.5
AX
-- OEM_REPOSITORY    5242880     5234688     5234688     8192  99.8
AX
Checking init parameters:
-------------------------------------------------------------------
-
---
Done.
```

Check the amount of free space against the total size of the tablespace. As long as there is more than 20% free space, there is no need to take further action (free space in percentage is shown in the second column from the right).

Since version 8, Oracle has been able to “auto extend” its data files up to a given maximum size, which is set during the creation of the data file. Maconomy normally creates data files with a maximum extent of 2 GB. This may cause some confusion, so if checkOracle reveals that a tablespace may be close to full, a second check should be made:

1. Log in to the system as “Maconomy”.
2. Start sqlplus.
3. Run the following check:

   ```
   select sum(bytes), sum(maxbytes), tablespace_name from
   sys.dba_data_files
   group by tablespace_name;
   ```

   This check will show you the maximum size of the tablespace, and how much it is actually using at the moment.

Example

```
maconomy 11 % sqlplus system/manager

SQL*Plus: Release 8.1.6.0.0 - Production on Thu Dec 4 15:48:12 2003
(c) Copyright 1999 Oracle Corporation. All rights reserved.

Connected to:
Oracle8i Release 8.1.6.0.0 - Production JServer Release 8.1.6.0.0 -
Production

SQL> select sum(bytes), sum(maxbytes), tablespace_name from
sys.dba_data_files group by tablespace_name;

SUM(BYTES) SUM(MAXBYTES) TABLESPACE_NAME
---------- ------------- ------------------------------
1918894080     4194304000  MACINDEX
```

```
2298478592      4194304000  MACONOMY
2097152000      0           MACROLLBACK
524288000             786432000   MACTMP
5242880         83886080    OEM_REPOSITORY
123731968             2097152000  SYSTEM

6 rows selected.
```

In this case, there is plenty of free space for the tablespaces to grow, with MACROLLBACK as the exception.

Check the Oracle Trace Files

Oracle writes errors to trace files, which should be checked weekly. To find the files, please check your server installation documentation to see where the `oraclebase` is located.

In that folder you will find the following path to Oracle's log files:

```
<oraclebase>/admin/<SID name>/log
```

If for instance `oraclebase` is `/data02/app/oracle`, and `<SID name>` is `macora`, the path will be

```
/data02/app/oracle/admin/macora/log
```

Entering `maconomy 30 % ls -l` may yield the following result:

```
-rw-r--r--      1 oracle    dba    1480142      04    dec    08:37
alert_macora.log
-rw-r-----      1 oracle    dba    524    20    aug
   10:11arc0_10516_macora.trc
-rw-r-----      1 oracle    dba    524    23    sep    08:00
arc0_10530_macora.trc
-rw-r-----      1 oracle    dba    524    03    nov    08:00
arc0_11790_macora.trc
```

In this case, there are three trace files and the database log file. Trace files should be checked by a Database administrator or by one of Maconomy's technical consultants.

Check the Operating System Error Log File

By checking the error log of the operating system, you can catch many errors before they become problems. On AIX use `errpt`; on Windows check the event viewer.

An example from AIX:

```
errpt
IDENTIFIER TIMESTAMP T C RESOURCE_NAME DESCRIPTION  4865FA9B
   1124152503 P H rmt0     TAPE OPERATION ERROR
4865FA9B 1124150003 P H rmt0     TAPE OPERATION ERROR
22006523 1029083903 T H rmt0     TAPE DRIVE FAILURE
2A9F5252 1010132303 P H tok0     WIRE FAULT
74533D1A 0923105203 U H SYSIOS   LOSS OF ELECTRICAL POWER
```

Check the Date of Export Files

If you are using the Maconomy backup script as described in the "Backup and Restore Guidelines" chapter of this manual, check the date in the Maconomy export file folder (typically

`MaconomyNT\Backup` on Windows). The date of the export files should be current; otherwise, something has gone wrong.

Run Cleaning Tape at the Tape Station

If a tape station is installed locally, a cleaning tape should be run once a week (for instance to avoid the error seen above). See also the bi-annual routine of cleaning the tape station.

You should also ensure that you can read what has been written on a random tape from last week's backup, and remember to follow the guidelines from the manufacturer of the tape about the lifetime of the tapes.

Regular Routines — Monthly

The routines described in this section should be performed monthly.

Reinstall Random Files from the Backup Tape

This check should be performed if you take a backup of more files than the databases data files and the database dumps generated by the Maconomy backup script.

To control the quality of the backup, the system administrator should reinstall some files from the backup tape. The restored files should be compared with the original ones. This check naturally only makes sense if there has been no change to the file since the backup was taken.

If you are using `tar` on Unix to write to the tape, like the Maconomy backup script does, please note that `tar` uses absolute paths and will overwrite the original file. So before a restore is made, make a copy of the files you are trying to restore.

Several programs can be used on the different platforms supported. On Windows, `CSDIFF` can be used. On AIX, the internal program `diff` can be used.

An example from AIX:

```
cd /data02/app/maconomy/w_8_0/MaconomyDir/Database mv Schema.cc.sql
Schema.cc.sql.orig
tar xv /data02/app/maconomy/w_8_0/MaconomyDir/Database/Schema.cc.sql
diff Schema.cc.sql Schema.cc.sql.orig
```

If the files are identical, the command `diff` will return no comments. `Diff` can be used for both binary and text files.

If there are no differences, you can delete the original file:

```
rm Schema.cc.sql.orig
```

The `tar` command will return this output when a file exists (for other backup programs, please refer to the manual or online help):

```
tar x /data02/app/maconomy/w_8_0/MaconomyDir/Database/Schema.cc.sql
```

As an alternative, all Maconomy installations contain a small command line program, which on all platforms is called `binsum`. You could call it twice, once for each file and compare the checksums. If the checksums are identical, so are the files.

Example:

```
maconomy 18 % binsum Schema.cc.sql
Checksum:      52508 Size: 802 KB      Schema.cc.sql

maconomy 19 % binsum Schema.cc.sql.orig
```

```
Checksum:      52508 Size: 802 KB      Schema.cc.sql.orig
```

Check that Backup Tapes can be Read on another Server

It is important to ensure that the backup can be read with another machine and with another tape drive, in case the original server is destroyed or stolen. This check can be done by completely restoring the system to the hard disk. Alternatively, restore some of the files and then read the complete contents of the tape.

Again please note that `tar` uses absolute paths, and it will overwrite any existing files.

Deposit Complete Backup in Safe Place

Every month you should deposit a complete backup in a fireproof safe deposit box or in the bank.

Import a Dump from the Backup to a Test Company

To ensure that the test data is up-to-date, a copy of the live data should be installed each month. If you are running some tests, this check can be postponed.

This is the guideline for doing this. You should only do this if you are the database administrator. We will use Maconomy's internal tools, since the syntax is mostly the same on all platforms.

The following default syntax is used:

```
<version>, <data_source>, <data_target>, <filename>
```

There are four steps to this:

1. Empty the target company.
2. Export the data from the source company.
3. Import the data into the target company.
4. Verify that you can log in to the target company.

Empty the Target Company

Before an import can be made, the target company must be emptied for data (truncated). This is done by using the `-ZT` parameter:

```
MaconomyServer.<version> -S<data_target> -ZT
```

For example:

```
MaconomyServer.w_8_0 -Snmtest -ZT
```

The output looks like this (note that the server version and database type may vary):

```
MACONOMY SERVER for ORACLE81 vers. 32.01.0.320148 compiled Sep 30
2003 10:23:13
: Import/Export.

Truncating ABSENCECALENDARLINE   Done. Truncating ACCESSCODELOG
   Done.
Truncating ACCESSINFORMATION     Done.
...
```

Ending with

```
Truncating WORKFLOWITEM    Done. TOTALS:
613 relations truncated.
```

Export the Data from the Source Company

Exporting data from the data source is done by using the `-E` parameter. This should not be done while there are users on the system:

```
MaconomyServer.<version> -S<data_source> -E<filename>.dbd
```

For example:

```
MaconomyServer.w_8_0 -Snm -Enm.dbd
```

The output looks like this (note that the server version and database type may vary):

```
MACONOMY SERVER for ORACLE81 vers. 32.01.0.320148 compiled Sep 30
2003 10:23:13
: Import/Export.

Database application version: Maconomy W 8.0
Exporting      ABSENCECALENDARLINE     .     Records     exported:
   0
Exporting      ACCESSCODELOG     .     Records     exported:   0
Exporting      ACCESSINFORMATION .     Records     exported:   1
Exporting      ACCESSLEVELHEADER .     Records     exported:   0
Exporting      ACCESSLEVELLINE   .     Records     exported:   0
Exporting      ACCOUNT     .     Records     exported:   0
Exporting      ACCOUNTBALANCE    .     Records     exported:   0
Exporting      ACCOUNTREFERENCE  .     Records     exported:   42
...
```

Ending with

```
Exporting      WORKFLOWITEM      .     Records exported: 0

TOTALS:
         613   relations exported.
         31130 records exported.
```

Import the Data into the Target Company

Import the data into the data target by using the `-I` parameter: `MaconomyServer.<version> -S<data_target> -I<filename>.dbd` For example:

```
MaconomyServer.w_8_0 -Snmtest -Inm.dbd
```

The output looks like this (note that the server version and database type may vary):

```
MACONOMY SERVER for ORACLE81 vers. 32.01.0.320148 compiled Sep 30
2003 10:23:13
: Import/Export.

Database application version: Maconomy W 8.0
Importing      ABSENCECALENDARLINE     .     Records     imported
   0
Importing      ACCESSCODELOG     .     Records     imported    0
Importing      ACCESSINFORMATION .     Records     imported    1
```

```
Importing       ACCESSLEVELHEADER .     Records     imported    0
Importing       ACCESSLEVELLINE   .     Records     imported    0
Importing       ACCOUNT     .     Records     imported    0
Importing       ACCOUNTBALANCE    .     Records     imported    0
Importing       ACCOUNTREFERENCE  .     Records     imported    42
...
```

Ending with

```
Importing       WORKFLOWITEM      .     Records     imported    0

TOTALS:
          613   relations imported.
          31130 records imported.
```

Check that you have imported as many tables and records in the target as you exported from the source.

Verify that You can Log in to the Target Company

If you have forgotten to truncate the data in the target company before the import, you will not be able to log in to the system.

Regular Routines — Biannually

Every six months you should perform the following routines.

Make a Complete System Restore

Restore a complete backup, log in to the restored system, and see if the integrity of the system is intact.

Clean the Tape Station

If the tape station is placed in a local server machine, twice a year (or quarterly) you should open the server box and clean it inside. This is important for reliable tape backups. For further instructions, please refer to the manual supplied by the manufacturer of the tape station.

Replace Backup Tapes

Every six months, or at least every year, you should replace your backup tapes to ensure reliable tape backups.

Checklist

The following list is a brief checklist of things to remember. You can make a copy of this page and hang it on the wall.

- Check the backup
- Check that the Daemons are running
- Check the temporary area for old files
- Check the Maconomy error log
  - Error types
- Check for free disk space

- Check for free space in the Oracle database
- Check the Oracle trace files
- Check the operating system error log file
- Check the date of export files
- Run cleaning tape at the tape station
- Reinstall random files from the backup tape
- Check that backup tapes can be read on another server
- Deposit complete backup in safe place
- Import a dump from the backup to a test company
  - 1. Empty the target company
  - 2. Export the data from the source company
  - 3. Import the data into the target company
  - 4. Verify that you can log in to the target company
- Make complete system restore
- Clean the tape station
- Replace backup tapes

Common Tasks

This section describes some tasks which are commonly performed on a Maconomy server.

Shutting down the Server (Windows)

If you want to shut down the server, and you want to be completely sure that the server is shut down fast, follow this procedure:

1. Log on locally as user `Maconomy`.
2. Stop the service MaconomyDaemon (type `net stop MaconomyDaemon` in a command prompt, or use the Services window).
3. Shut down all processes called `MaconomyServer.exe`.
4. Stop the service OracleService`<SID>` (type `net stop OracleService<SID>` in a command prompt, or use the Services window).
5. Shut down the server.

Who is Logged in to the Maconomy System?

To find out who is logged in to a given Maconomy system, follow this procedure:

1. Log in locally as Maconomy.
2. Open a command prompt.
3. Run the MaconomyServer process associated with the Maconomy system.

   On Linux there is a symbolic link to the server executable in the '/data/maconomy/bin' folder that includes the system name.

   On Windows the '-i' option must be used to identify the system configuration to use.

In addition the '-S' option must be used to specify which database shortname to connect to. The '--users' option can then be used to list the current user session for the system.

Examples:

MaconomyServer.exe -i<system-name> -S<shortname> --users
macoracle.<system-name> -S<shortname> --users

In the table listing you can see the user name and the client type being used as well as creation time and last-access time of the user session.

Change the Password for the Maconomy User (Windows)

If you wish to change the password for the Maconomy user, you can do so by logging on as the local system administrator. However, note the following:

- All services installed with Maconomy run as the Maconomy user, so it is necessary also to change the password for these services. In some cases, some of the Oracle services may also run on the Maconomy user account, and the password should also be changed here. The Maconomy services include:
  - `MaconomyDaemon`
  - `WebDaemon.<shortname>.<port>`
  - `MscriptDaemon<portal id>`
- The Maconomy backup runs as an `at` job. Normally all `at` jobs are set up to run as the Maconomy user in "Scheduled tasks" under "Advanced options." If other Maconomy jobs are running, these should also be checked.
- On the day following a password change, it should be checked that the backup has run correctly, and that all Maconomy related services are running.

Frequently Asked Questions

This section contains a description of the most common errors encountered on a Maconomy Server. You can use the hints in this section to find and correct an error yourself, or to trace an error before creating a support case with Maconomy Customer Support Service.

No Users are able to Log in to the System

If no user is able to log on in the morning or after a reboot of the server, you should check the following before contacting Maconomy CSS:

- Can you log on to the Maconomy server? If not, there may be a hardware problem or a faulty network connection.
- Is MaconomyDaemon running?
- Is OracleServiceORCL running?
- Is there a shortage of disk space? Perform the routines described in "Regular Routines — Daily."
- Is a backup process hanging, blocking the Maconomy system? Check your backup. Activate the DOS window with the backup process, and press CTRL+C to break the operation. Then start the Maconomy system.
- On systems that are running MS SQL Server, the system may be set in single user mode.

  To resolve this, use the SQL Server administration program to set the number of connections to unlimited (0).

- On UNIX systems, the folder `FontSupport` may have been renamed to `FontSupport.DontUse` to prevent users from logging in during backup. Rename the folder back to `FontSupport`.
- On three-tier systems, is the `TNSLISTENER` service running on the database server?

I am the Administrator, and I have Lost my Password

If you are the Maconomy administrator, and you forget your password to the Maconomy application or you enter a wrong password three times in a row, Maconomy shuts you out of the system. To remedy this, an SQL update must be performed. Contact Maconomy CSS.

To reduce the likelihood of this happening, assign administrator rights to a trusted super-user of the Maconomy system. This is done in the Users window in the Maconomy Set-Up module. Then, if either of you forget the password, the other can open up the account again without involving Maconomy CSS.

The Users Keep Experiencing Client Crashes

If users experience client crashes when updating data in the system, you can check the following:

- Check the log files as described in “Check the Maconomy Error Log” and “Check the Operating System Error Log File.”
- Is the database tablespace running low? Perform the routines described in “Check for Free Space in the Oracle Database.”
- Is the database inconsistent as a result of an error or a bug? Contact Maconomy CSS.
- Are the users’ clients compatible with the Maconomy TPU? Check the TPU release notes or contact Maconomy CSS.

The System is very Slow

If you experience a lag in the system’s response, consider the following:

- Has a user started an Analyzer report without parameters? This can cause a considerable drain on server resources. See “Killing a Process in Oracle” in this manual for information about killing the process in question (terminating the client is not sufficient).
- Has a user initiated a huge print job, for example, printing all G/L entries for a year? This will cause a system slowdown.
- Is it around the end of a month or the beginning of a month? Sometimes many tasks need to be taken care of at the same time, such as invoicing, posting, reporting, and approving time sheets. Consider changing your routines to even the server load, or upgrade the server hardware to match the maximum load.
- Sometimes, Portal users experience a loss of connection to the web server during such peak times. This can be resolved by increasing the server time-out period as described in “Maconomy Server Options and Parameters.”
- Does the Maconomy server initialization file (`MaconomyServer<shortname>.I`) contain the option `-d`? That option turns on server debug information and can reduce overall system performance. It should only be on during the installation phase.
- Does the file `TestMasks` exist in the `MaconomyDir` directory? That file, which is issued by Maconomy R&D, is used in some support incidents to run dynamic tests on the system. This will reduce overall system performance. After completion of the tests, the file is not needed any more and should be removed.

- Is something else (other than Maconomy) using large amounts of system resources? On Windows, open Task Manager to see current processes. On UNIX (AIX), use the following shell command: `ps -e -o "%C %a"` to check the CPU usage.
- For more information about performance, see “Performance” in this manual.

Performance Tuning, Debugging, and Logging

On a running Maconomy installation, it may be required to measure and tune components. The two older user interfaces, the Java client and the Portal, have dedicated debug and tuning guidelines, whereas the newer components can be analyzed and monitored end-to-end, which gives a more complete view of where to adjust and optimize. The following sections apply to the older clients; a dedicated section for the end-to-end Performance Monitor follows them.

Java Client Debug Options

In the Java client, you sometimes need to perform diagnostics and testing. For this reason, a number of features have been built into the interface, but hidden away so that the general user is not confused by the options.

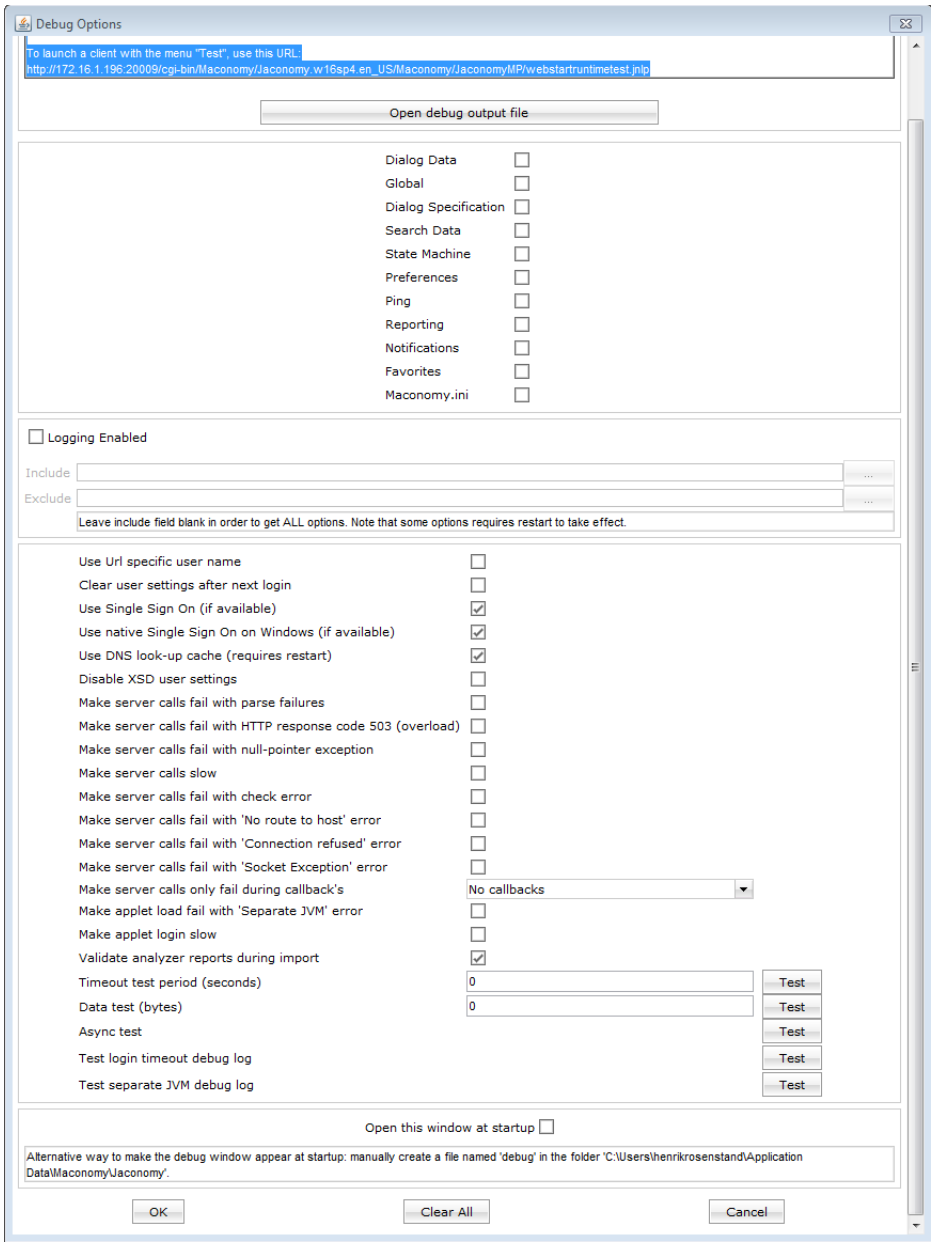
This chapter describes these options. It is, however, important to note that Deltek does not guarantee that all options will remain available in the client. Deltek also reserves the right to add and remove options and features in service packs without notice. The following description is a snapshot of the Debug Options window.

Open the Debug Options Window

To open the Debug Options window, complete the following steps:

1. Launch a Java client.
2. Click **About Maconomy…** in the Help menu.
3. Press CTRL+D.

The following window is displayed.



This window contains the following buttons: **OK**, **Clear All**, and **Cancel**. Clicking **Clear All** resets the window to the default settings: nothing is logged, but some options are set.

General Information

The top pane of the window contains information about the output of the debug information and about how you can launch the client with a Test menu from a browser.

Output

The debug information of the type or types that you select in the Options pane is output in two places:

- A text file called debuglogfile.txt in the Maconomy folder in the Application Data folder of the current user, for example, C:\Documents and Settings\<username>\Application

Data\Maconomy\Jaconomy\debuglogfile.txt. This text file is reset (truncated) every time that you restart the client.

- The data is also displayed in the Java console. You can display the Java console by opening the Java control panel (either from Windows or Macintosh Control Panel or by typing javaws at a system prompt and then choosing Edit » Preferences). In the Java control panel, click the Advanced tab and select Show console in the Java console node of the Settings tree. You must do this before you start the Java client, or you must restart the client.

In the following, whenever mention is made of the client log or logging, the log entries go to the debuglogfile.txt file and the Java console.

Test Menu

By specifying an alternative URL in a browser, you can launch the Java client in a mode that includes a new menu called GUI Test. To specify the alternative URL, replace the last section of the regular URL, webstart.jnlp, with webstartruntimetest.jnlp. The new menu contains two options for testing the system. The tests are performed by Maconomy engineers to ensure the integrity and the memory management of the client.

- **Open All Windows** — If you select this option, the Java client opens and closes all windows in the system, one by one. The test might also be performed if many new window layouts are installed in the system, to ensure that they are all working.
- **Open All Reports** — If you select this option, the Java client opens and closes all reports installed in the system, one by one.

Debug Levels

The Java client potentially outputs a large amount of log data. In fact, all communication with the server can be logged. To reduce the amount of logged information, and so that you can pinpoint your diagnostics, you have the option to only output debug information of certain categories.

If any of the debug levels are selected, every window in Maconomy gets a wide, yellow border on the left side of the window to remind you that debug output logging is active, as shown in the following figure.

The debug levels are described in the following.

- **Dialog Data** — If you select this field, the Java client logs dialog data information. In addition to the raw data, the dialog data specifies a number of structural things about the dialog that you are opening, such as the state of the various actions, is New Upper permitted, and so on.
- **Global** — If you select this field, the Java client logs global information. A substantial amount of global information is registered when you log in to the client:
  - Version information
  - The special Java client dictionary
  - The server Maconomy.ini file

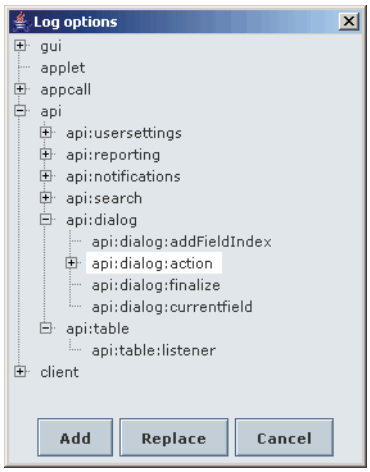
- Access information
- Menu structure
- Pop-up literals
- Shortcut keys
- Web server and web services information
- The location of the online help file
- The state of the menu tree

- **Dialog Specification** — If you select this field, the Java client logs dialog specifications. Dialog specifications consist of the MDL layout, a specification of the fields and actions that are available in the dialog, information about the size and location of the dialog, and other information
- **Search Data** — If you select this field, the Java client logs search data. Search data consists of the search specification that is sent to the server and the result of the search that is returned from the server.
- **Preferences** — If you select this field, the Java client logs preferences data. When you save preferences in the Preferences window, the settings are communicated to and acknowledged by the server.
- **Ping** — If you select this field, the Java client logs the client's keep-alive signal (ping). This signal is sent by the client to prevent the server from automatically logging the user out and subsequently losing high-level locks. By default, a ping is sent every 270 seconds.
- **Reporting** — If you select this field, the Java client logs reporting data. The reporting data is similar to search data, but applies to data that is sent from and to reports in the Reporting Component.
- **Notifications** — If you select this field, the Java client logs any communication related to retrieving and setting notifications. The notifications are refreshed at login, when you select Refresh Notifications, or when you perform certain actions in the client with which a workflow is associated, and also once every 300 sections (5 minutes).
- **Favorites** — If you select this field, the Java client logs all calls to favorites.
- **Maconomy.ini** — If you select this field, the Java client logs all parameters/values read from the Maconomy.ini file.

Logging Code Flows

In the third pane of this dialog, you can choose to log the execution of all or some types of code in the Java client. You can do this by selecting one or more trigger points, which indicate when a log entry should be written to the output file and the Java console. Note that you should only do this if a Maconomy support consultant or other Maconomy representative instructs you to do so. A log entry records the code that is run, including any parameters and calls to the Maconomy API, when a trigger point of the selected type is called (by performing certain actions in the client).

- **Logging Enabled** — When you select this field, Maconomy logs the code flow according to the trigger points that you selected in the Include field. Logging only occurs—and you can only include and exclude trigger points—when this field is selected.
- **Include** — In this field, you select which trigger points should lead to log entries. When you click the Browse button to the right of the field, the following window opens.



Here, you can browse to the appropriate trigger points. When you highlight a trigger point and click the Add button, the trigger point is added to the **Include** field. If you click the Replace button, the **Include** field is cleared before the trigger point is added. Note that you can include an entire hierarchy of trigger points by adding a trigger point with a plus sign to the left. For instance, you can log all of the code that is used when working with dialogs by selecting the trigger point **gui:dialog**, or you can choose to look only at the code that is executed when a progress bar is shown by adding the **gui:dialog:progressbar** trigger point.

You can also choose to enter the trigger points manually. If you leave the field blank, all trigger points are selected, leading to large amounts of data being logged. Note that some trigger points require that the Java client is restarted before any code execution is logged.

- **Exclude** — In this field, you can select one or more trigger points to exclude from the trigger points that you selected in the **Include** field. For instance, if the **Include** field contains **gui:dialog**, you can choose to exclude the code that is executed when working with combination boxes in dialogs by entering **gui:dialog:combobox** in this field. The trigger points are selected in the same way as for the **Include** field.

Various Options

The fourth pane of the Debug Options window lists various options.

- **Use URL-specific user name** — If this field is selected, it is easier to run different Java clients, for example, one that points to a test server and one to the production server. If this field is not selected, the Java client suggests the same user name when launching either client. This field should normally be selected.
- **Clear user settings after next login** — If this field is selected, user settings such as window placement, menu tree state, and so on are cleared the next time that you log in to Maconomy. Note that the selection of this field is also cleared so that you do not accidentally clear user settings every time that you log in to Maconomy.
- **Use Single Sign On (if available)** — If this field is selected, the client attempts to log in automatically using single sign-on (SSO). Note that selecting this field does not turn SSO on; enabling SSO is a complex process that involves the installation of key authentication software on the server. The reason that this option exists is that if you use the autostart feature of the Maconomy client for the Java platform, you cannot choose whether you want to log in automatically or not. This field should normally be selected.

- **Use native Single Sign On on Windows (if available)** — The default value is on because the Java single sign-on is not accessible from Java programs on Windows.
- **Use DNS look-up cache (requires restart)** — If this field is selected, the client attempts to resolve the URL that is used for server communication (changes it to an IP address) by looking it up in the DNS (domain name system) lookup cache that is maintained by the Java framework (JVM). This is the default, and the standard Java way of operation.

  However, in some setups, the web server IP address may change periodically. As a result, if you allow the operating system to go into hibernation mode after you launch the client, you might be unable to reconnect to the Maconomy system after your computer has left hibernation mode. This is because the client resolves the URL to the IP address that is found in the DNS lookup cache, but the IP address has changed.

  In such situations you can deselect this field to let the Java framework resolve the URL every time that you log in to the Maconomy system. However, there is a security issue involved in this, because it opens your system to DNS spoofing attacks. You should only deselect this field if you are certain that your system is otherwise protected from such attacks. Note that deselecting this field corresponds to setting the property networkaddress.cache.ttl to the value 0 in the file <Your Java installation>\jre\lib\security\java.security.
- **Disable XSD User Settings** — This disables the use of XSD-based user settings. User settings are saved using the "old" format that does not support certain new features like "Favorites" in "Find" panes. This is normally only used for testing backward compatibility of the Java client user settings
- **Make server calls fail with parse failures** — This makes web server requests fail with “parse” failures, that is, the Java client/applet receiving information from the web server that was not expected from the web server request. This is normally only used for testing failure handling in the Java client/applets.
- **Make server calls fail with HTTP response code 503 (overload)** — This makes HTTP requests from the Java client/applets fail on the web server with HTTP response code 502. This is normally only used for testing failure handling in the Java client/applets.
- **Make server calls fail with null-pointer exception** — This makes web server requests fail with a null-pointer exception in the Java client/applets. This is normally only used for testing failure handling in the Java client/applets.
- **Make server calls slow** — This makes web server requests “slow” in that all web server requests are delayed by 3 seconds. This is normally only used for checking performance of the Java client/applet UI.
- **Make server calls fail with check error** — This makes web server requests fail with a simulated CheckError(...) Maconomy application error. This is normally only used for testing error handling in the Java client/applets.
- **Make server calls fail with ‘No route to host’ error** — This makes web server requests fail with a simulated No Route To Host exception. This simulates network connections that fail because the physical connection is lost or other conditions prevent the network from connecting to the host. This is normally only used for testing failure handling in the Java client/applets.
- **Make server calls fail with ‘Connection refused’ error** — This makes web server requests fail with a simulated Connection refused exception. This simulates network connections that fail because the port is not available or is protected by a firewall. This is normally only used for testing failure handling in the Java client/applets.
- **Make server calls fail with ‘Socket Exception’ error** — This makes web server requests fail with a simulated Socket Exception exception. This simulates an error in the

underlying protocol (TCP). This is normally only used for testing failure handling in the Java client/applets.

- **Make server calls only fail during callbacks** — This makes web server requests only fail during certain callback types. This simulates errors and failures that happen during callbacks. The callback type can be selected in a pop-up. The callback types selected are "No Callbacks," "All callbacks," "Only progress bar callbacks," and "Only progress bar 50%+ callbacks." This is normally only used when testing error and failure handling during callback handling in the Java client/applets.
- **Make applet load fail with 'Separate JVM' error** — This makes the Java applet loading process in the Maconomy Portal fail with a Separate JVM error. This is normally only used when testing Java applet browser/Java compatibility for new versions of browsers and Java.
- **Make applet login slow** — This makes the Java applet login process slow (30 seconds). This is normally only used when testing timeout errors in Java or browsers.
- **Validate analyzer reports during import** — This makes the Java client/applets validate .xro Analyzer reports when imported. This is normally selected; deselect it only if certain old .xro Analyzer reports need to be imported that seemingly have no access.
- **Timeout test period (seconds)** — This makes web server requests take a specified number of seconds. The user needs to be logged in before this test can be used. The number of seconds is specified in a field next to the check box. This is normally used to test for the presence of timeouts in network setups. For example, when running Analyzer reports that take a long time it is important that network timeouts do not terminate the Analyzer report server requests. Clicking the "Test" button initiates the test.
- **Data test (bytes)** — This makes web server requests send a specified number of bytes. The user needs to be logged in before this test can be used. The number of bytes is specified in a field next to the check box. This is normally used to test for the presence of network traffic shaping lowering the priority of "large" HTTP requests. Clicking the "Test" button initiates the test.
- **Async test** — This performs an asynchronous web server request. Normally the Java client/applet only performs one web server request at a time. This is normally only used for testing the communication between the Java client/applet and the web server.
- **Test login timeout debug log** — If Java client/applet debug logging has been enabled on the web server, this creates and sends a debug log timeout message to the web server. This can be used to check that the web server debug logging setup is working.
- **Test separate JVM debug log** — If Java client/applet debug logging has been enabled on the web server, this creates and sends a Separate JVM message to the web server. This can be used to check that the web server debug logging setup is working.

Open at Startup

In the fifth pane of the Debug Options window, you can choose to open this window at startup.

- **Open this window at startup** — If this field is selected, the Debug Options window is opened before the login window is shown. In this way, you can select the appropriate debug options or set code flow trigger points before you log in.

  Instead of selecting this field, you can do the following before you start the client:

  1. Open the Maconomy\Jaconomy folder in the Application Data folder of the current user, for example:

     ```
     C:\Documents and Settings\<username>\Application
     Data\Maconomy\Jaconomy
     ```

2. In this folder, create an empty file called debug.
3. Start the client normally.

The Java client will now start up, showing the Debug Options window before the login window.

Portal Performance

This section is an introduction to the Portal, with an emphasis on the basic technology behind it.

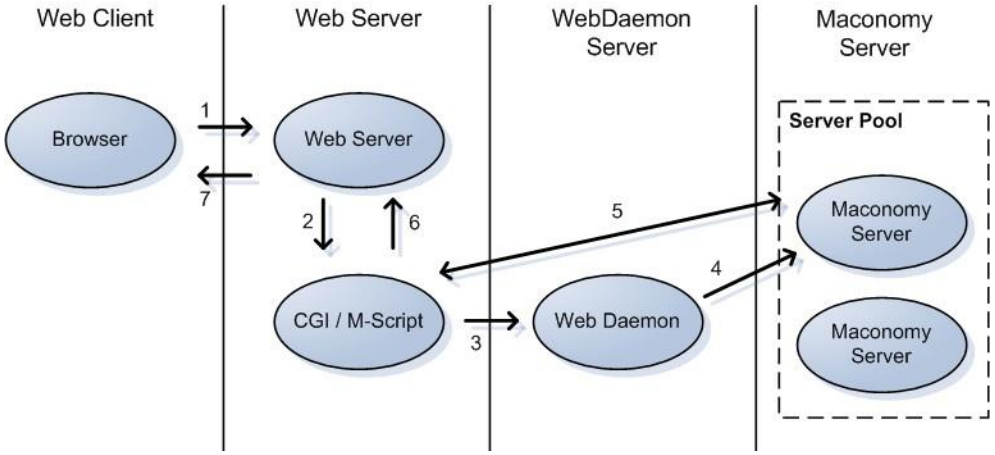
From a user's point of view, the Portal consists of a browser interface to Maconomy. A user enters data into Maconomy, or queries the Maconomy database, for example to generate reports and dashboard overviews. The data is then saved or displayed, respectively.

From a technical point of view, the Portal is a complex framework of advanced technologies, all of which need to be synchronized for optimal performance.

Communication Overview

The following communication model describes what happens when you request data from Maconomy in the Portal—that is, when your browser communicates with a Maconomy server.

The numbers in this figure refer to the explanation that follows it.



1. The flow of communication is initiated by a web browser as in the preceding example or by the Maconomy client for the Java™ platform.

   The browser points to an IP address (your link to the Portal) that is hosted by the web server in this diagram and requests a page (which is really an M-Script program).

2. The web server executes the M-Script program from its CGI directory. The web server also retrieves graphics and fixed HTML elements.
3. The M-Script program requests data from Maconomy. The request is passed to a process— called the WebDaemon process—that runs on the Maconomy server.
4. The web daemon finds an available Maconomy server process (or spawns another server process, if necessary), and passes the request for communication from the M-Script program to the server process.
5. When communication is established, the M-Script program can exchange data directly with the Maconomy server process.

Performance Pitfalls

In the preceding diagram, you can identify several dangers to Portal performance: the data transitions from browser across the network to the web server, from M-Script to the web daemon, and from the web daemon to the Maconomy server. In addition, the Maconomy server communicates with the underlying database (Oracle, IBM DB2, or MS SQL Server) and back across the network again.

Hence, when a user complains about poor Portal performance, the problem is often not related directly to the Portal, but to the communication between the browser and the server. “Maconomy Performance Monitor” in this manual provides a description about how to clock the actual server transactions by analyzing SQL queries, dialog actions, and so forth.

The complex nature of web interfaces makes it important to tune each element in the chain of communication to each other. The following section describes these elements one by one.

Portal Applications

It is important to note that the prime objective of the Portal is to provide an easy-to-use, accessible, and elegant interface to the Maconomy database. The Portal is designed to streamline operations that can be complex to perform in the standard Maconomy client, such as the Procurement flow. To accomplish this, the Portal developers assembled parts from different Maconomy windows on tabs in the Portal components. However, the basic Maconomy functionality is not changed—behind the scenes, Maconomy still has to open several windows (such as the Requisitions, Requests for Quote, Purchase Orders, and Document Archives windows, in the case of the Procurement flow) to process the information that is put into the Portal. Thus, when measuring the performance of the Portal, you should compare it to the time that a similar workflow would take in the standard Maconomy client.

HTML was originally designed to display static text and graphics. It has since been enhanced to allow for dynamic text—called Dynamic HTML, or DHTML—but because of the nature of HTML, some processes are not suitable for the Portal. This includes processes that involve large-scale manual entry into tables (such as the General Ledger). Updating cells in a table in DHTML is simply too slow. Therefore, some processes are better done in the standard Maconomy client (which works over the Internet, too, if that is a requirement).

The Browser

The browser is the software used for interacting with the Maconomy database, through the Portal framework. This section describes various issues related to browser set-up and recommended guidelines for setting up the browser.

All that a browser does is to display text according to a set of rules. These rules are fixed in the standard called HTML. This presents a performance issue in itself, because to render complex structures such as a Maconomy tab/sub-tab workspace, an enormous amount of text commands must be sent to the browser, along with the data. This is the same as when surfing the Internet—the more complex the page, the longer you have to wait for it to load.

Adding to this issue is the problem of interpreting the HTML. Different browsers interpret HTML in different ways. Some are lenient and forbearing; others interpret the rules for HTML strictly and give errors if they are not followed. The same brand of browser might interpret HTML differently depending on the platform and version—for example, Internet Explorer for Windows 5.0 is not the same as Internet Explorer 5.0 for the Macintosh or Internet Explorer 6.0 for Windows. Finally, all browsers have errors. This makes it very hard to write perfect HTML, and makes it necessary to reduce the number of browsers that are certified for running the Maconomy Portal. Currently, Maconomy supports Internet Explorer 5.5 and above and Mozilla 1.3.1.

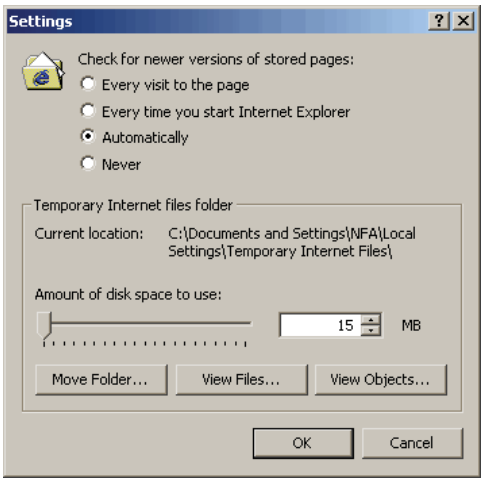
Because Maconomy has no control over the browser setup (for security reasons), it is important that you configure the browser for optimal performance by following the guidelines described in the following sections.

Cache Settings

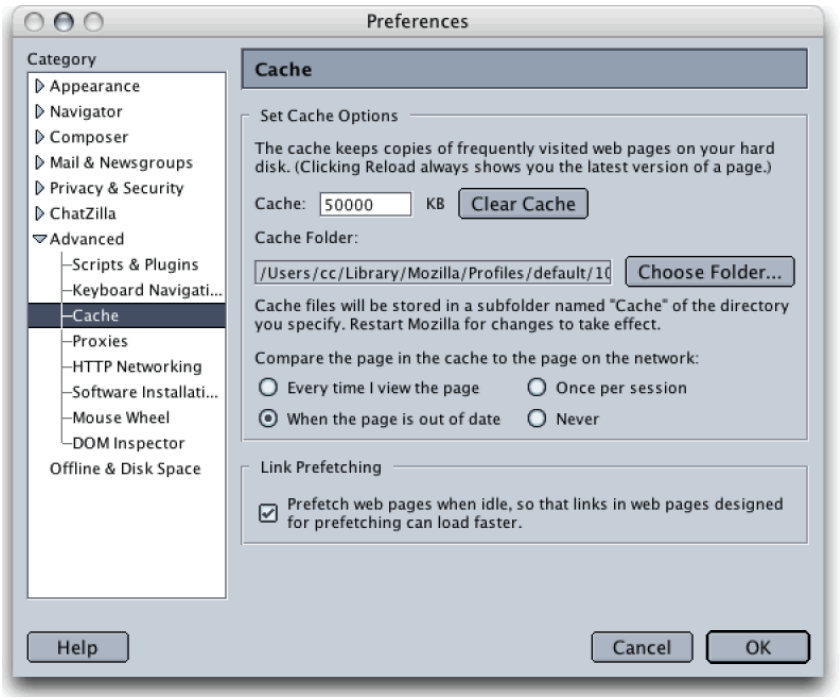
The cache is a location on your hard disk where web pages and files (such as graphics) are stored as you view them. This speeds up the display of pages that you frequently visit or have already seen, because the browser can open them from your hard disk instead of from the web.

Thus, instead of downloading your company logo, for example, each time that you open the Portal, the browser gets the logo from the hard disk instead, which increases performance. It is therefore important that the cache functionality is turned on, and that you have sufficient cache space.

In Internet Explorer, this is done by opening the Internet Options window in the browser Tools menu. In the General tab, click the Settings button in the Temporary Internet files island. This displays the following window.



In Mozilla for the Macintosh, you adjust the cache in the Preferences window.



Make sure that sufficient space is allocated to the cache. No less than 12 MB is recommended. The more that you browse the Internet, the larger the cache that you need, because different web pages are added to the cache, and whose eventually push out the files that were cached from the Portal.

Deleting Cache on Exit

In some companies, corporate policy dictates that the browser deletes all files in the cache when the browser exits. This means that the cache must be rebuilt every time that you open the Portal, and this makes the Portal appear slower.

Compatibility Issues

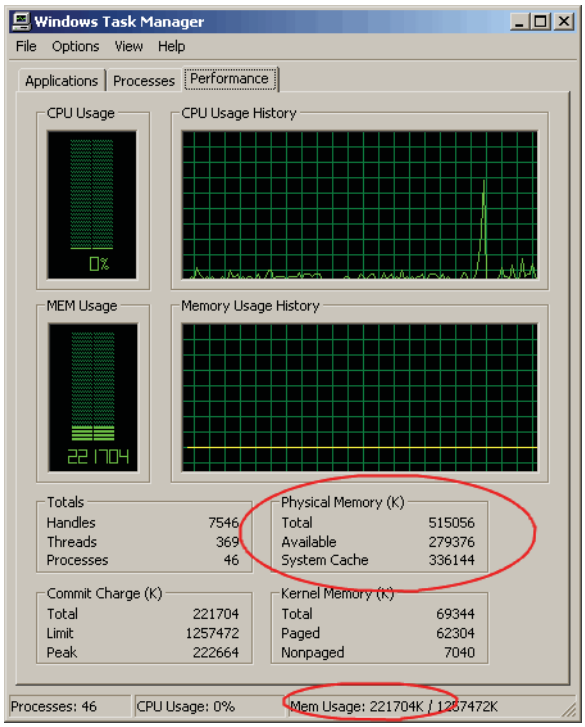
Some non-Maconomy web applications are very sensitive to the use of browser caching. The application night fail to display updated pages if the browser caches pages from the application. Turning off the browser cache seems to solve this problem. However, turning off the browser cache is not the solution, because doing so creates performance problems for all other applications. Instead, the reason for the cache-related errors should be found on the web server. Usually, adjusting the HTTP settings of the web server (for example, indicating that certain pages are always “expired”) solves the issue without impairing the performance of other web applications.

Hardware

A critical factor related to Portal performance is the hardware that is used for running the browser. When the browser loads a page in the Portal, a large amount of data is transferred. This is because the HTML DOM (Document Object Model) is very complex, especially in Portal components with many lines and columns in the sub-tab. Loading such a page requires many resources for the browser.

Memory

In Windows, you can check whether your browser is running out of resources by loading a complex page in the Portal, then opening the Task Manager on the Performance tab.



At the bottom of the screen, check the value for **Mem Usage**. If the first of the values listed there exceeds the value for total physical memory as displayed in the Physical Memory (K) island, Windows is swapping memory to your hard disk. This causes heavy disk activity (“disk thrashing”) and poor performance.

The solution is to install more memory in the system.

On the Macintosh, the problem can be resolved by assigning more memory to Mozilla. If that is not possible, more memory must be installed.

Processor

When the browser receives the page, it must render it—the browser takes the HTML DOM objects and places them on the page according to the HTML rules. With complex pages, the process of rendering is very dependent on processing power. That means that installing a faster CPU can dramatically improve browser performance.

Background Images

The Portal framework uses themes to maintain a common layout of the Portal components. Sometimes background images are used to apply an attractive backdrop to web pages. However, it turns out that some browsers are very sensitive to the use of background images, especially when many tables and frames are used, as is the case in the Portal.

Therefore, if performance is a problem, Deltek recommends that you exchange the background images for a single color. The performance gain can be quite dramatic.

Note that as of Maconomy 8.0 Service Pack 16, the default “green theme” does not contain background images.

Network Considerations

The network can also have an impact on performance. When opening a Portal window, Maconomy gives priority to transferring as much data as possible when opening the window. This means that the initial load time might be relatively long, but the time that it takes to update or change data in the window is relatively short. If the network transport mechanism is slow, opening a Portal window will seem very slow.

Bandwidth Latency

When accessing the Maconomy server over the Internet (and not the local area network (LAN)), Portal performance is dependent on the quality and speed of your Internet access. If you are using a satellite connection, for example, there is a certain amount of latency—which you experience as a lack of responsiveness, a short delay from clicking a button in the Portal, for example, until you can see that anything happens.

There is no resolution to this issue, except for getting a more efficient connection.

Network Component Errors

In some cases, when the Portal seems to perform poorly, it is because a network component (a network interface card, a router, or a cable) is broken. In such cases, the actors in the communication between browser and Maconomy might need send and resend the same communication many times before it is received.

To resolve this issue, or to rule it out, you can monitor your network using special network monitoring software or instruments.

Compression

Some network components support the use of compression; that is, the network data packets are compressed before they are transmitted and decompressed when received. The idea is that when the network packets are made smaller, they transfer faster. In a normal LAN setting, the overhead of the compression/decompression routine cancels the positive effect of the compression, because a LAN is usually very fast. However, when working over the Internet (such as when using VPN), compressing data can improve performance.

Proxy Servers

In situations where Internet access is very slow or unstable, you can install a proxy server. A proxy server is a device that uses idle bandwidth to download (mirror) entire often-used web sites and keeps them updated. Users then connect to the Internet through the proxy server. When a user tries to reach one of the mirrored sites, the proxy server serves the web pages that are stored locally on the proxy server.

You cannot, of course, mirror the entire Maconomy system, but you might find that a proxy server can reduce the time that it takes to load the Portal framework files (graphics, scripts, static HTML pages, and so on).

M-Script

In the remaining part of the communication route (web server—M-Script—web daemon—Maconomy server), there are only a few performance pitfalls.

Web Server

When the web server receives a request from the web client (the browser), it processes the request by launching an M-Script program. This is a CPU-intensive process, and it is important that the web server machine is configured properly, with a powerful CPU and access to sufficient

memory and disk space. For more information about this, see the documentation for your web server.

Authentication

A web server usually uses authentication when serving files to a browser. This means that for every file served, the web server checks that the recipient of the file is in fact the one who requested the file. This is for security reasons and helps to prevent unauthorized use of the network.

However, a browser always tries to receive a file anonymously. This results in the following sequence of events:

1. The web browser requests a file anonymously.
2. The web server requests authentication (an ID).
3. The web browser tells the web server who is calling by submitting an IP address.
4. The web server decides whether it is OK to serve the file to the browser and acts accordingly by serving the file or denying access.

For “safe” files (such as the Portal theme files, company logo, static HTML, and so on), you can reduce the workload of the server and the network by specifying in the web server setup that those files do not require authentication.

Notes on Specific Web Servers

Supported web servers are Apache and Microsoft Internet Information Server (IIS). For Apache, Deltek recommends version 2.x. Do not use version 1.3.27.

For IIS 2003, note that M-Script from TPU 55 is required because of a bug in IIS.

Web Daemon

When the web daemon receives a request from the M-Script program, it connects the M-Script program with a Maconomy server process. If no server process is available, the daemon might spawn another process. With regard to performance, it is important that the number of server processes is kept under control.

The maximum number of server processes that are spawned by the web daemon is controlled by the web daemon option ServerProcs. If the number is very low, you risk that M-Script programs must wait in queue before being served by a Maconomy server process. On the other hand, if the number is very high, you risk overloading the CPU and database of the server machine.

As a rule of thumb, set the value of ServerProcs to 3 or 4 per CPU in the server machine. See also “Web Daemon Options and Parameters” in this manual.

Maconomy Server

If the Portal still seems sluggish after you have verified the following:

- The browser, web server, and web daemon are all configured correctly
- Both the web client machine and the servers have sufficient resources
- There are no network errors

it is time to look at the operations that happen on the Maconomy server.

Debugging M-Scripts

M-Script contains facilities for logging server usage and information about functions, parameters, and timing.

When the M-Script interpreter is started, it reads its parameters from a file named `<name of executable>.I` in the same directory as the executable. Thus, if the name of the executable is `maconomymscript.exe`, the parameters are found in the file `maconomymscript.I`. Into this file you can add an entry called "log," with which you can specify the areas that you want to log.

The possible values for the log entry are as follows.

Log Entry	Value
none	No logging at all (default)
sql	Log only `maconomy::sql` calls.
analyzer	Log only `maconomy::analyze` calls.
login	Log only `maconomy::login` calls.
functions	Log all `maconomy::` calls.
scripts	Log script execution load and time information.
serverstat	Log information about server-side timing.
query	Log information about received query variables.
postdata	Log all incoming POST data in a file with the same name as the M-Script executable appended with a `.postdata.log`.
cleanup	Log information about session cleanup activities.
localization	Log information about processing of localization dictionaries.
GUI_windowStatus	Log information about opening and closing of GUI windows.
GUI_warnings	Log extra information about the GUI.
client	Combines `script`, `query` and `localization`.
server	Combines `sql`, `analyze`, `login`, `functions`, and `serverstat`.
GUI	Combines `GUI_windowStatus` and `GUI_warnings`.
all	Log all of the above.
performance	Log information about how various parts of M-Script perform.

Log Entry	Value
tab	Organize log entries in a tabular format.

When logging to analyze M-scripts for performance reasons, it is recommended that you use the following log file setup:

```
log=scripts;server;tab
```

This logs information about the load time and run time of scripts; calls to the server with SQL, Analyzer commands, login requests, what M-Script functions are performed, and server timing, and output the log file in a tabular format, which is ready for further processing.

Assume that you want to analyze what actually takes place when you open the Portal. You want to see which scripts are run, and what data is selected on the server. To do this, it is recommended that you follow these steps.

1. Create a Portal instance by copying the M-Script executable and its associated initialization file on the web server and pasting it in the same location, but using a different name.

   For instance, if the name of the M-Script is `maconomymscript1.exe`, you can copy and paste it with the name `maconomymscript1.log.exe`. Remember to copy the associated `.I` also.

   The sole purpose of the new Portal instance is to log specific actions. The reason for creating another Portal instance is that when you turn on M-Script logging, M-Script logs every activity within the scope defined by your `Log` entry in the `.I` file. If many users are working with the Portal, the log quickly becomes very large. When you create another instance, it is used only by you, and you can thus control which actions are logged.

2. In the new `.I` file (`maconomymscript1.log.I`), add the entry

   ```
   log=scripts;server;tab
   ```

3. In the browser, access the Portal by using the new URL from the preceding example (`<your webserver>/maconomymscript1.log.exe`). In the web server directory, a log file is created called `maconomymscript1.log.log`.

4. You can now open this file in an editor, or (better) import it into a spreadsheet application for analysis.

   The information in the file tells you which scripts were run, how long they took, and what data was selected in the database. Based on this information, you can pinpoint problems in M-Script code or SQL code.

For more information, see the M-Script Reference manual, which contains an explanation of the individual columns in the log file.

Debugging Using the Maconomy Performance Monitor

The Maconomy Performance Monitor (MPM) is a debugging tool that is designed to time the different processes that happen on the Maconomy server.

One instance of the MPM system is initiated inside every MaconomyServer process that is running on the Maconomy server machine. The MPM system monitors and times much of the internal functionality in the Maconomy server, such as actions, SQL expressions, MQL requests, Universe Report requests, and so on. The MPM system is controlled remotely by the Coupling Service. The number of items to measure and monitor can be specified in configuration files

placed on the Maconomy server. MPM measure-points (probes) can be active, inactive, or conditional (only logging when a certain condition is true).

All output from the MPM system is written in XML formatted text files. These XML files can be analyzed and interpreted afterwards to gain the relevant information.

Often MPM is used for monitoring processes running for a “long time,” for example, monitoring all SQL expressions that run for more than 5 seconds. In this way you can monitor some features of a test installation setup, for example.

See “Maconomy Performance Monitor” in this manual for more details.

Implementation-Specific Issues

Browser and web server setup, network, and bandwidth are all important factors when considering Portal performance, and it is relatively simple to tune performance by adjusting these elements. However, another factor that influences Portal performance is more elusive—the specific implementation of the Portal in your company.

The following sections examine some of the parameters that affect performance in the Portal.

Size of Data Sets

With the Job Budgeting component as a starting point, this section describes a number of areas that affect the size of the DOM object that must be loaded into the Portal.

Creating the Budget

Usually, the budget is created when the job is created. A job is created from a template. A standard budget is typically assigned to this template. Sometimes a template budget can contain hundreds of lines. Thus, even though creating the job is done by simply selecting a job type and clicking a button, it can take a long time for Maconomy to perform the operation because of the many behind-the-scenes operations of copying budget lines, deriving job parameters, and so forth.

A budget that has hundreds of lines is a very large DOM object. Therefore, it takes a long time to render it in the browser (especially if the Portal theme contains background images). In addition, it often turns out that only a few budget lines are actually used.

From Service Pack 16, sub-tab can be shown in increments and read-only. This means that the browser gives control back to the user after displaying a small amount of lines in the sub-tab, and the user clicks a button to see the next set of lines. In addition, the user must click the line to open it for input. This is all done to increase performance in tables.

Therefore, you can optimize Portal performance just by creating more job templates, each of which has a different (much smaller) set of budget lines.

Using the Budget

Favorite values (pop-ups that show the last ten tasks used, for example) are convenient. However, you should be aware that adding a favorite pop-up to a field in a sub-tab increases the size of the DOM object considerably. Thus, if performance is a problem, consider using the standard CTRL+G search instead of favorite pop-ups.

Evaluate carefully which columns you actually need to show in the component. Is it really necessary to show both the task description and the activity description? Is it at all necessary to show the activity? Each column adds to the size of the DOM object. Therefore, you should remove unnecessary columns from the component layout.

Job Setup

As of Maconomy 8.0 Service Pack 16, you can specify which fields in the Job Information card cause job parameters to be rederived. Using this functionality dramatically reduces the time that it takes to update a job.

Searching

In the standard client, the default number of items returned in a search is 30. To see more items, you click the **More** button. In the Portal, 30 items may be too high—reduce the number by setting the relevant options in the search configuration files.

Server Load

Having regular peaks in the Maconomy server load is normal. The peaks are a result of the company's business practices. For instance, if all project managers invoice customers at month's end, a peak load will occur at month's end.

It is helpful to consider these peaks when starting resource-intensive processes on the server. For instance, it is not a good idea to start a complex report on Monday morning when everybody is submitting their time sheets. If the Finance department needs to produce weekly reports, do it on Wednesdays instead of Mondays, and so on.

Reevaluating the timing of business processes is a very individual matter—it varies from company to company. The Maconomy Performance Monitor can help you identify peak server loads.

Performance Monitor

The Maconomy Performance Monitor (MPM) is a framework for measuring time-critical code and for structured reporting of important performance information. The MPM framework was originally implemented in the MaconomyServer 26.03 series released with Maconomy 7.0 in TPU 32.

In Maconomy 2.2 and 2.1.1 CU2 (SP4), a similar framework is introduced in the new Java server platform (the Coupling Service). In the 2.2 EA release, the framework is available only as a stand-alone tool for the Coupling Service, while in 2.2 GA and 2.1.1 CU2 the MPM framework in the Coupling Service has been extended to also include time measurements carried out on the Maconomy Server. Currently, only functionality in the Maconomy Server and Coupling Service can be logged (that is, no M-Script information is available).

The framework contains a set of predefined time measuring objects, called probes, placed in, for example, the code of the MaconomyServer. Each of these probes is an object dedicated to a specific monitoring task. An example of a task is measuring the execution time of MSL dialog scripts. Besides measuring time, a set of attributes are assigned to every probe.

MPM v. 1.1.6, which is described in this document, is released with Maconomy 2.1.1 CU2 and 2.2 GA. Several probes are introduced after the initial release in TPU 32, and the description of available probes within the Maconomy Server is based on the functionality available in the abovementioned releases.

The probes can be configured and initialized on a runtime system to monitor performance or system activity. The output is logged in XML format. Using XSL transformations, the structured output information can be used to extract useful reports on system load and behavior.

MPM in the Coupling Service

This section describes MPM as a combined performance-reporting tool for the Coupling Service and the Maconomy Server.

In addition to the absence of the MPML specification language, the main difference in the way the MPM framework is configured in the Coupling Service compared to the Maconomy Server is that when the MPM framework is enabled in the Coupling Service, all probes and all probe attributes are also enabled by default. This makes it easier to do ad-hoc performance reporting without having to write a detailed MPML configuration in order to get the relevant probes logged—assuming that you even know what the relevant probes are.

If appropriate, filters can be specified to restrict which probe entries are included in the performance reports, but this is mostly considered relevant if you want to run with the framework enabled for an extended period of time. Unlike the Maconomy Server MPM framework, it is not possible to control which attributes within each probe entry are reported; if a probe is included by the filters, it is always reported with all available attributes. Therefore, the XML reports generated by the Coupling Service MPM framework are typically wordier and larger than the XML reports generated by the Maconomy Server MPM framework.

Setup

The MPM framework in the Coupling Service can be configured through a number of settings in **server.ini** in the Coupling Service configuration folder. Changes to these settings are automatically picked up by the running Coupling Service, and the MPM framework is dynamically reconfigured accordingly.

If XML reporting is enabled—which happens by default when the MPM framework is enabled—an output folder with a unique name (**mpm-<YYYYMMDD>-<hhmmss>**) is created in the specified XML output location. An output folder is created every time the Coupling Service is restarted and whenever the MPM configuration changes.

In the XML output folder, a new XML report file (**<session-name>-<YYYYMMDD>-<hhmmss>-<serial-no>.mpm.xml**) is written for each request executed by the Coupling Service. If probe entry filtering is configured, only reports with at least one included probe entry result in an XML report file.

By default, the MPM framework in the Coupling Service will only collect probe entries from the Maconomy Server if these are explicitly enabled. This is done by adding a default MPML configuration file ('MPMDefault.cfg') to the server definitions folder, as described in the section 'MPM in the Maconomy Server' below. Alternatively, all server probes can be enabled from the Coupling Service by setting 'mpm.server.all-probes = true'. However, having all server probes enabled will negatively affect performance and memory consumption of the Maconomy server, so by default this is turned off.

The following table describes the settings used to control the MPM framework:

Setting	Description	Default Value
mpm.enabled	Specify whether the Performance Monitor framework is enabled.	false
mpm.xml.enabled	Specify whether XML reporting is enabled (when **mpm.enabled** is set to **true**).	true
mpm.xml.location	Change the output location where XML reports are saved.	${maconomy.log.directory}/mpm/xml

Setting	Description	Default Value
mpm.server.all-probes	Whether all MPM probes on the Maconomy server should be collected.	false
mpm.filters.<element>	Set the filter expression that should be applied to the specified element.	n/a
mpm.time-unit	Specify the time unit to use when filtering time durations. All unique prefixes of any of the following units are valid: NANOSECONDS, MICROSECONDS, MILLISECONDS, SECONDS, MINUTES, HOURS, DAYS	SECONDS

Filtering

Filtering can be performed on the elements listed in the following table.

Element name	Type	Description
session-wallclock-time	time	The total wall-clock time duration of the monitoring session. This filter either includes or excludes all probe entries in the session.
session-user-time	time	The total user-level CPU time duration of the monitoring session. This filter either includes or excludes all probe entries in the session.
session-proc-time	time	The total process CPU time duration of the monitoring session. This filter either includes or excludes all probe entries in the session.
Session-name	string	The name of the active Performance Monitor session.
probe-name	string	The name of the probe that produced the probe entry.
title	string	The title of the probe entry.
principal-name	string	The principal names of the current user. For example, a Maconomy user principal name looks like this: **<db>.<locale>\<user-name>**.
arg	string	The string representation of the arguments passed to the probe entry, if it represents a function call.
key	string	The attribute keys set on the probe entry. The use of attributes on probe entries is entirely probe-dependent.
value	string	The attribute values set on the probe entry. The use of attributes on probe entries is entirely probe-dependent.

Element name	Type	Description
wallclock-time	time	The wall-clock time duration of the probe entry.
user-time	time	The user-level CPU time duration of the probe entry.
cpu-time	time	The total CPU time duration of the probe entry.

Filter Expressions

A small expression language is available for creating filter expressions.

The syntax of a filter's expression is a logical combination of one or more condition expressions. The logical combinators which can be used are the following:

- <expr1> `or` <expr2>
- <expr1> `and` <expr2>
- `not` <expr>

There are two types of condition expressions:

- string-expr: String comparison expressions.
- number-expr: Number comparison expressions.

The syntaxes of these condition expression types are:

- string-expr: [`contain`(`s`)|`match`(`es`)] [regex] <pattern>
- number-expr: [<|<=|=|>=|>] <value>

Examples

Filter Expression	Comments
`contains "abc"`	The value must contain the string **abc**.
`matches regex \d+`	The value must consist entirely of digits.
`> 10`	The value must be greater than 10.
`<= 200.0`	The value must not be greater than 200.
`matches foo or contains bar`	The value must either match the string **foo** or contain the string **bar**.
`not < 25 and not > 50`	The value must be between 25 and 50. Note that **not** must be used twice because it only applies to the nearest sub-expression.
`not < 25 and <= 50`	Same as the preceding expression, with a reverted comparison operator instead of the second negation.

Using MPM

The MPM framework in the Coupling Service can be used in either of two ways:

- To investigate a known performance problem.
- To monitor a system in order to discover performance problems.

The difference between these use cases is primarily the filtering that should be applied before the framework is enabled.

Investigate a Known Performance Problem

When collecting a performance report on a known performance problem, you generally want to restrict reporting to the specific scenario that you wish to investigate. This means that you want to narrow the filtering as much as possible to keep the number of generated reports to a minimum. Therefore, you probably want to apply one or more of the following filters:

- The principal name of the user performing the action. If you know the Maconomy user name of the user, the following filter expression restricts reporting to just that user (assuming the user name is sufficiently unique within the system):

  ```
  mpm.filters.principal-name = contains "<user-name>"
  ```

- A lower bound for the runtime of the action that you want to investigate. A well-chosen lower bound ensures that reports are not generated for faster requests executed by the same user:

  ```
  mpm.filters.session-wallclock-time = >= <time-duration>
  ```

  If the problem is known (or assumed) to consist of a single operation with bad performance and the report should only contain the probes leading to that operation, the following filter can be specified instead:

  ```
  mpm.filters. wallclock-time = >= <time-duration>
  ```

  Make sure the time unit matches the time unit specified by **mpm.time-unit**. The default time unit is seconds.

Monitor a System to Discover Performance Problems

In this scenario, a system is known (or suspected) to display bad performance, and you want to collect performance reports on these issues for all users that might be affected. To do this, you probably want to apply one or more of the following filters:

- A lower bound for the runtime of the action that you want to investigate. A well-chosen lower bound ensures that reports are not generated for “normal” requests:

  ```
  mpm.filters.session-wallclock-time = >= <time-duration>
  ```

  If the problem is known (or assumed) to consist of a single operation with bad performance, and the report should only contain the probes leading to that operation, the following filter can be specified instead:

  ```
  mpm.filters. wallclock-time = >= <time-duration>
  ```

  Make sure that the time unit matches the time unit specified by **mpm.time-unit**. The default time unit is seconds.

- A tell-tale name, title, argument value, or property key/value. If the performance problem is known (or suspected) to occur within a specific probe, and you already have tell-tale information about the probe entries that you are interested in, you can use advanced filtering to get reports on only those probe entries. This is what the following filter elements are for:

```
probe-name, title, arg, key, value
```

MPM Output

If XML report generation is enabled (by the **mpm.xml.enabled** option), a new XML report file is generated in the current output folder under the XML output location (specified with **mpm.xml.location**). Only performance monitor sessions with one or more probe entries included by the probe entry filters result in a new XML report file.

The format of these XML report files follows the same MPM schema as the reports that are generated by the Maconomy Server MPM framework, and they can be transformed to human-readable formats using the same XSL and XSL-T transformation tools.

MPM in the Maconomy Server

This section describes MPM as a stand-alone performance-reporting tool for the Maconomy Server.

Setup

The MPM is part of the Maconomy server and server tools package from TPU 32 and later. Hence, the following only applies to installations using TPU 32 or later.

The MPM Framework

The Maconomy installation must be prepared for MPM before using the logging functions.

To prepare your system for using the Maconomy Performance Monitor, complete the following steps:

1. Create a log destination folder for the XML output files. On Unix/Linux systems, remember to grant the Maconomy user write access to the folder.
2. Insert an entry in the ApplicationConfig section of a custom version of the server's Maconomy.ini file, indicating where the output log folder is placed. Add a line similar to the following:

   ```
   [ApplicationConfig]
   MPMLogDirectory=D:/tmp/log
   ```

   If this option is not present, or if the folder is not accessible or cannot be created, the MPM functionality is turned off completely.

   The folder path can contain a number of format placeholders that are expanded with values from the current date and time. In this way, the log folder can be periodically "rotated" to limit the number of files written into one folder (which is a known source of filesystem performance problems on NTFS partitions).

   The format placeholders consist of a **%** followed by a single, case-sensitive character. The following table describes the supported format placeholders.

Placeholder	Format	Comment
`%Y`	YYYY	4-digit year
`%m`	MM	2-digit month
`%d`	DD	2-digit day of month

Placeholder	Format	Comment
%H	HH	2-digit hour (24-hour clock)
%I	HH	2-digit hour (12-hour clock)
%M	MM	2-digit minute
%S	SS	2-digit second
%p	AM/PM	AM/PM suffix (for 12-hour clock)

For example, the following folder path results in a folder being created every day:

```
MPMLogDirectory=D:/tmp/log/mpm-%Y%m%d
```

3. Place the following configuration file in the server **Definitions** folder:

   `MPMDefault.cfg`

   This file contains configuration settings described in MPML (Maconomy Performance Monitor Language). The information in this file overrules hardcoded probe defaults in the Maconomy software.
   On Unix/Linux platforms, remember to grant the Maconomy user read access to the file.
   Unlike earlier Maconomy releases, the "user-configuration" file `MPMUser.cfg` is no longer in use and can be omitted.

Third-Party Setup

Make sure that your Internet browser can perform XSL transformations (for old versions of Internet Explorer, the XSL extension msxml3 may need to be installed). If your browser cannot perform XSL transformations, the browser will not display any meaningful data.

For large XML files, it could be preferable to do the XML-to-HTML translation using a stand-alone translator such as Microsoft msxsl, and then view the resulting HTML file in the browser.

Probes

Probes are predefined objects that measure time. The probes are placed in the Maconomy Server code by Maconomy R&D, in places where structured log and timing functionality could be useful at runtime. In addition to measuring time, the probes can have several attributes assigned to them. For instance, an MSL script probe can contain the number of database gets executed inside the script. Attributes are of the type Integer, Float, String, Time, Date, and Boolean.

All probes and attributes are described in MPM Probe Descriptions.

A probe can be in one of the following states:

- **Close** — The probe and all of its attributes are closed and inactive.
- **Register** — The probe and its active attributes are registering—that is, they are not logging, but information is gathered for the probes on the actual stack (that is, embedded probes). This information can be logged if the probe is triggered either by a condition or an embedded probe.
- **Log** — All active attributes are logging.

An attribute can be in one of the following states:

- **Off** — The attribute is closed and inactive.
- **On** — The attribute is on, and information may be registered and logged, depending on the probe state.

A probe can be conditional, that is, a state change occurs if the condition is met. The condition is a function of one or more of the probe attributes (for example, calling depth is more than 4, or the time spent is more than 30 seconds.).

For example, a probe can be in the register state, but if the time spent is more than 30 seconds, the probe goes into log state.

Probe states, attribute states, and probe conditions are specified in the configuration files using MPML. For detailed information on MPML, please see Configuration using MPML.

Probes can be embedded within each other. For instance, inside the probe-timing of a Universe Report execution, other probe-timings of SQL executions can exist. Generally, the order of embedding is defined at run time. This means that you generally cannot predict the order of the embedding of probes in your reports.

On the other hand, when you know that several database operations are needed to execute an analyzer report, you could use this knowledge in your Analyzer Report using the Analyzer and SQL operations probes.

Using MPM

As of 2.3 GA, the MPM framework can only be enabled from the Coupling Service. See the section MPM in the Coupling Service for more information.

MPM Output

This section describes the output produced by the MPM.

Output Files

The output files are placed in the output log folder as indicated in the server Maconomy.ini file. Any file cleanup must be done manually.

All standard log files are named MPM-log <Username>#<pid> <time>. Snapshot log files are named MPM-snapshot <Username>#<pid> <time>.

The files are all in XML format, and can thus be transformed using standard XSL transformations using browsers or standalone XML translators. All tags that are used exist in the XML namespace http://www.maconomy.com/mpm, and thus the XSL reports must access the tags inside that namespace.

The size and relevance of the output files depend on the probes and attributes that are specified in the configuration files. The amount of data logged, as well as the purpose and the types of reports to be run, should be carefully considered when specifying the probe configuration files.

Output Analysis

When the MPM system is turned off, or when some Maconomy Server processes have ended, you can start analyzing the XML output files. Because the output files are written in a standard XML format, you can use any XML editor/viewer (such as a browser) to view the files in a nicely formatted way.

It is also possible to translate the XML data into nice-looking reports using third-party XML tools, for example, using XSL-T stylesheets or other kinds of XML parsing tools.

To use a report it is essential that the information needed by the report is logged in the output files, so the configuration files should be carefully considered with respect to the information needed and the amount of data collected.

XSL Report Examples

A small demonstration report suite is supplied with the MPM framework. The reports are XML reports that transform the output log files to useful information using XSL.

Reports can be created using XSL and the description of the specific probes and their attributes. To create reports, knowledge about the embedded probes and log configuration requirements is essential.

To transform a logged MPM XML file to HTML using one of the demo suite XSL stylesheets, an XSL translator tool must be applied, for example, the Xalan tool, which is included in the TPU, or any other XSL translator tool, getting both the XML file and XSL file as input.

Using Xalan, the command line option looks like this:

```
Xalan -o myoutputfile.html mympmlogfile.xml myxslreportfile.xsl
```

Note that Xalan currently does not accept certain characters in filenames, such as #. Descriptions of the example reports follow.

MSL Subprogram Script Report

Reports Timing MSL Subprogram Scripts when Executing MSL Functions

Subprogram scripts are MSL scripts that are executed when calling MSL functions (for example, Library Scripts) inside MSL programs. Example calls could be CREATEJOURNAL or POST_JOURNAL.

These probes are mainly of interest to R&D developers.

Log Configuration Requirements

The MSLSubprogramScript probe and the Title, TotalGets, TotalGetNexts, TotalPuts, ProcessTime, and ProcessTimeTotal attributes must be logged for optimal report information.

MSL Statistics Report

Reports Timing Total Rankings for MSL Subprogram Scripts when Executing MSL Functions

Subprogram scripts are MSL scripts that are executed when calling MSL functions (for example, LibraryScripts) inside MSL programs. Example calls could be CREATEJOURNAL or POST_JOURNAL.

These probes are mainly of interest to R&D developers.

The report is a summation of total time spent and total number of times invoked for every dialog and subprogram script.

Log Configuration Requirements

The MSLSubprogramScript probe and the Title, TotalGets, TotalGetNexts, TotalPuts, ProcessTime and ProcessTimeTotal attributes must be logged for optimal report information.

MQL Execution Report

Reports Timing MQL Query Executions, Including Information about Universe Reports

The executions are considered the time spent fetching data in the report, including database access.

Log Configuration Requirements

The MQE/ROE probes can be used, but especially ROECreate, MQECompileParseUniverse, and ROEPrint, and the Title, WallclockTime, and WallclockTimeTotal attributes must be logged for optimal report information. In addition, UniverseId/ReportId should be logged on the appropriate probes.

Configuration Using MPML

This section describes how to set up performance monitors in the MPM configuration file using the Maconomy Performance Monitor Language (MPML).

MPML Examples

MPML is a simple language for specifying constraints on probes. MPML is used in the configuration file MPMDefault.cfg.

Each file can contain specifications addressed to multiple users. For each user section in a file, multiple probes can be configured.

For example:

```
/* MPMDefault.cfg */
MPML 1 // MPML Version tag

Client windows macintosh// Only for windows and macintosh clients Company
dk60demo      // Only for company dk60demo (db. shortname)

/** Address all users ************************/ Users #All
// Log if more than 30 secs are spent probe AnalyzerProbe register
register -> log if (WallClockTime > 30.0)

// Turn logging of attributes on attributes    Title on      // (Default is off)
ProcessTime on ProcessTimeTotal on WallClockTime on WallClockTimeTotal on

/** Address user Administrator specifically **/ User Administrator
probe  MSLSubProgramScript log attributes      Title on
TotalGets on TotalGetNexts on TotalPuts on ProcessTime on ProcessTimeTotal on

probe ParseDSProbe close

// Address users MOK, TSP and OS process #4912 specifically
/*****************************************/ Users MOK, TSP, #5912
probe  MSLDialogScript log attributes    Title on
ProcessTime on ProcessTimeTotal on

probe AnalyzerProbe log attributes all on
probe  MSLSubProgramScript log attributes      all on
Count off CountAll off
```

Syntax Definition

The following set of code is the formal MPML syntax definition in BNF (Backus-Naur form):

```
start
: versionDefinition
| versionDefinition probeSpecifications
;

start
: versionDefinition clientDeclaration companyDeclaration
| versionDefinition clientDeclaration companyDeclaration probeSpecifications
;

probeSpecifications
: probeSpecification
| probeSpecifications probeSpecification
;

probeSpecification
: userDeclaration probeDeclarations
;

versionDefinition
: /* empty */
| MPML version
;

version
: REAL
| INTEGER
;

clientDeclaration
: /* empty */
| CLIENT clientIdentifiers
;

clientIdentifiers
: ALL
| specificClientIdentifiers
;

specificClientIdentifiers
: ID

| specificClientIdentifiers ID
;
```

```
companyDeclaration
: /* empty */
| COMPANY companyIdentifiers
;

companyIdentifiers
: ALL
| specificCompanyIdentifiers
;

specificCompanyIdentifiers
: ID
| specificCompanyIdentifiers ID
| STARTUP    // Note: STARTUP literal prefixed by '#', i.e. '#STARTUP'
| specificCompanyIdentifiers STARTUP
;

userDeclaration
: USERS userIdentifiers
| USER userIdentifiers
;

userIdentifiers
: #ALL // Note: ALL literal is prefixed by '#', i.e. '#ALL'
| specificIdentifiers
;

specificIdentifiers
: specificIdentifier
| specificIdentifiers specificIdentifier
;

specificIdentifier
: peopleIdentifier
| processIdentifier
;

processIdentifier
: '#' INTEGER
;

peopleIdentifier
: ID_
| INTEGER
| #STARTUP   // Note: STARTUP literal is prefixed by '#', i.e.
// '#STARTUP'
;
```

```
probeDeclarations
: probeDeclaration
| probeDeclarations probeDeclaration
;

probeDeclaration
: probeDefinition probeState probeAttributes
;

probeDefinition
: PROBES probeNames
| PROBE probeNames
;

probeNames
: probeName
| probeNames probeName
;

probeName
: ID_
| STRING
;

probeState
: /* empty */
| defaultState
| defaultState conditionedStates
;

defaultState
: stateType
;

stateType
: LOG
| CLOSE
| REGISTER
;

probeAttributes
: /* empty */
| ATTRIBUTE attributeDefinitions
| ATTRIBUTES attributeDefinitions
| ATTR attributeDefinitions
;
```

```
attributeDefinitions
: attributeDefinition
| attributeDefinitions attributeDefinition
;

attributeDefinition
: attributeName attributeState
;

attributeName
: ID_
;

attributeState

: ON
| OFF
;

conditionedStates
: conditionedState
| conditionedStates conditionedState
;

conditionedState
: stateType ARROW stateType ifCondition
;

ifCondition
: IF conditions
;

conditions
: '(' conditions ')'
| condition
| andCondition
| orCondition
| notCondition
| embeddingCondition
;

condition
: attributeValue operation attributeValue
| attributeValue operation interval
;

andCondition
: conditions AND conditions
```

```
;

orCondition
: conditions OR conditions
;

notCondition
: NOT conditions
;

embeddingCondition
: EMBEDDEDIN probeName
;

attributeValue
: constType
;

operation
: EQ_
| NOT_EQ
| LT_
| GT_

| GT_EQ
| LT_EQ
;

constType
: STRING
| ID_
| REAL
| BOOL_
| INTEGER
;

interval
: openingBracket
intervalOperand ';' intervalOperand closingBracket
;

openingBracket
: '['
| ']'
;

closingBracket
: '['
```

```
| ']'
;

intervalOperand
: INTEGER
| REAL
;
```

MPM Probe Descriptions

All probes are written to the log file as XML elements, and MPM attributes are written as XML elements that are embedded in the corresponding probes. The probe element has one XML attribute, the name attribute, which indicates the name of the probe class.

An example of an MSLDialogScript probe and an MSLSubProgramScript probe can be configured like the following in the MPMDefault.cfg file:

```
...
probe  MSLSubProgramScript log attributes      Title on
TotalGets on TotalGetNexts on TotalPuts on ProcessTime on ProcessTimeTotal on

probe  MSLDialogScript log attributes    Title on
ProcessTime on ProcessTimeTotal on
...
```

The MPM log file content that results from this configuration looks like the following:

```
...
<Probe Name="MSLDialogScript">
<Title>SYSTEM</Title>
<Probe Name="MSLSubProgramScript">
<Title>GETUSERINFORMATION</Title>
<TotalGets>2</TotalGets>
<TotalPuts>0</TotalPuts>
<TotalGetNexts>0</TotalGetNexts>
<ProcessTimeTotal>0.016</ProcessTimeTotal>
<ProcessTime>0.016</ProcessTime>
</Probe>
<ProcessTimeTotal>0.031</ProcessTimeTotal>
<ProcessTime>0.015</ProcessTime>
</Probe>
...
```

The following sections describe the probes and their (MPM) attributes. This information is valuable for configuring probes in the configuration files and for MPM log file analysis. All MPM attributes are implemented as embedded XML elements (that is, not as XML attributes).

Generic

Generic attributes are currently part of all probes. The information that these attributes contain can be extracted from all probes.

Attribute	Type	Description
`Title`	String	Probe instance specific title, for example, dialog name, and so on
`SerialNumber`	Integer	Supplement to the title
`MemoryUsed`	Integer	Memory consumption; unsupported
`Start`	Date	Start time and date
`End`	Date	End time and date
`Depth`	Integer	Stack depth of nested probes of this same type
`CountAll`	Integer	Number of times this probe class has been initiated since last command activate
`Count`	Integer	Number of times this probe class has been initiated since last command start
`EndedByUser`	Boolean	Indicates whether session was ended by user or forced to shut down by application
`ProcessTime`	Time	Process time spent in this probe instance; on Linux/Unix systems, process time may differ substantially from the wall clock times due to I/O operations.
`ProcessTotalTime`	Time	Process time spent in this probe instance and embedded probes; on Linux/Unix systems, process time may differ substantially from the wall clock times due to I/O operations.
`WallClockTime`	Time	Wall clock time spent in this probe instance; on Linux/Unix systems, process time may differ substantially from the wall clock times due to I/O operations.

Attribute	Type	Description
WallClockTotalTime	Time	Wall clock time spent in this probe instance and embedded probes; on Linux/Unix systems, process time may differ substantially from the wall clock times due to I/O operations.

Maconomy Performance Monitor

This is a Root element, and includes information that is specific to this MPM instance.

Attribute	Type	Description
Version	Real	MPM version number

Main

This is a Root element, and includes process and other MPM information that is specific to this process instance.

Attribute	Type	Description
Snapshot	Boolean	Indicates if probe dump a snapshot, that is, probe stack dump
ProcessId	Integer	OS process ID
ProcessName	String	Name of application MPM instance. For example, MaconomyServer 6.1
ProcessId	Integer	OS process ID
ProcessName	String	Name of application MPM instance. For example, MaconomyServer 6.1
ProcessVersion	String	Version of application MPM instance
UserName	String	Client username
TimeUnit	String	Time unit; unsupported
ClientType	String	Client type. For example, Macintosh, Windows, Java, WebDaemon, Batch Job, MScript, Server

Attribute	Type	Description
DatabaseShortName	String	Short name of database

MSLDialogScript

This refers to the probing execution of MSL dialog scripts (for example, NEWIN_UPPERPANE, ENTER_UPPERPANE).

The following scripts are executed when changing state in the dialog handler state machine.

For dialogs:

- NEWIN_UPPERPANE
- IN_UPPERPANE
- NEWENTER_UPPERPANE
- ENTER_UPPERPANE
- DELETE_UPPERPANE

For tab/sub-tab dialogs:

- NEWIN_LOWERPANE
- IN_LOWERPANE
- NEWENTER_LOWERPANE
- ENTER_LOWERPANE
- DELETE_LOWERPANE

An environment script exists for every dialog (for example, POSTING_ENVIRONMENT).

For prints, an environment script also exists, and a script for every block script in the MPL layout. The generic attribute title is the dialog script name (for example, TIMESHEETS_IN_UPPERPANE).

Attribute	Type	Description
ScriptdId	Integer	Script identifier; strictly internal information
ScriptName	String	Script name; unsupported
DialogKind	Integer	Internal type

MSLSubProgramScript

This refers to the probing execution of MSL subprogram scripts (MSL functions). Examples are CREATEJOURNAL and POSTJOURNAL.

Subprogram scripts are MSL functions and library scripts, mainly of interest for Maconomy R&D developers. The Title generic attribute is the subprogram script name (for example, CREATEJOURNAL).

Attribute	Type	Description
ScriptId	Integer	Script identifier
ScriptName	String	Script name
DialogKind	Integer	Internal type
TotalGets	Integer	Total number of cursor Get in the script and in embedded scripts
TotalGetNexts	Integer	Total number of cursor GetNext in the script and in embedded scripts
TotalPuts	Integer	Total number of cursor Put in the script and in embedded scripts

GetDS

This measures the time spent fetching the DialogSpecification. The generic attribute title is the dialog name (for example, Timesheets).

Attribute	Type	Description
DialogName	String	Dialog name
UseDDL	Boolean	Indicates if DDL is used; unsupported
FoundInCache	Boolean	Indicates if DS is found in cache; unsupported

ParseDSProbe

This measures the time spent parsing the DialogSpecification.

Attribute	Type	Description
DialogName	String	Dialog name
UseDDL	Boolean	Indicates if DDL is used; unsupported

DbCursorDeclareProbe

These are database cursor declaration functions.

Attribute	Type	Description
CursorId	Integer	Unique cursor identifier; no IDs reused
SqlQuery	String	Cursor select query; same query logged in the DbSQLFunctionProbe; only one of these attributes should be logging

DbSQLFunctionProbe

These are direct database SQL functions. The generic attribute title is the function type (for example, Commit, Open, or Fetch).

Attribute	Type	Description
CursorId	Integer	Unique cursor identifier; no IDs reused; matches with already declared ID
SqlQuery	String	Cursor select query; same query logged in the DbCursorDeclareProbe; only one of these attributes should be logging

AnalyzerProbe

This measures the time spent running an Analyzer report, including all database lookups, and so on. The generic attribute title is the internal Analyzer report name (for example, glentry.grf).

Attribute	Type	Description
ReportExternal Name	String	External name (title) of report

ClientCommProbe

This measures the time spent communicating with (also waiting for) the client. The generic attribute title is the communication direction, that is, Receive or Send

Attribute	Type	Description
ExternalId	Integer	External communication ID; only used in Receive probes
FunctionId	Integer	Communication function ID; only used in Receive probes
BufferLength	Integer	Length of buffer sent to/received from client

ROECreate

This is the ROE Probe that measures the time that is spent on creating a report object.

Attribute	Type	Description
ReportId	String	Name of report

ROEPrint

This is the ROE Probe that measures the time that is spent on printing a report object. Included in time are query execution and print execution.

Attribute	Type	Description
ReportId	String	Name of report

MQECompile

This is the MQE Probe that measures the overall time that is spent on compiling a query.

MQECompileParseQuery

This is the MQE Probe that measures the time that is spent on parsing a query.

MQECompileParseUniverse

This is the MQE Probe that measures the time that is spent on parsing a universe.

Attribute	Type	Description
UniverseId	String	Name of universe

MQECompileEnvironmentBuild

This is the MQE Probe that measures the building of the query/universe environment.

MQECompileResolve

This is the MQE Probe that measures the resolution of the object graph for a query.

MQESQLGeneration

This is the MQE Probe that measures the generation of native SQL.

MQESQLExecution

This is the MQE Probe that measures the execution of the native SQL and the building of XMIRDO.

MPLCompileAndMerge

This is the MPL Probe that measures the compilation and merge of XMIRDO and MPL.

MPLMerge

This is the MPL Probe that measures the merge of XMIRDO and MPL.

MPLOutput

This is the MPL Probe that measures the output of XMIRDO and MPL.

Attribute	Type	Description
`OutputType`	String	Output type generated; values for output type are HTML, PDF or name of export transformation

RPCCallFunction

This is the RPC Probe that measures the time that is spent executing Remote Procedure Calls (RPC).

Attribute	Type	Description
`FunctionName`	String	The name of the function being invoked.
`Arg1`	String	A textual representation of the first argument to the function (if applicable)
`Arg2`	String	A textual representation of the second argument to the function (if applicable)

SMLockWrite

This is the Shared Memory Probe that measures the time that is spent with write access to the shared memory segment.

SMLockBegin

This is the Shared Memory Probe that measures the time that is spent obtaining access to the shared memory segment.

SMLockEnd

This is the Shared Memory Probe that measures the time that is spent releasing access to the shared memory segment.

Automation and Customization

Most customer systems require automations and/or customizations. The following sections some of the tools/possibilities for customized solutions.

Workflow Spider

This document describes scripting in Maconomy using Workflow Spider.

Using the Maconomy Workflow Spider add-on products, you can create scripts that execute Maconomy reports and external programs, either from a menu or during the printing of standard printouts.

Introduction to Workflow Spider

With Maconomy Workflow Spider scripts, you can extend the standard Maconomy application. The Workflow Spider scripting language contains a powerful command type:

- Execute Report

  This command can execute a Maconomy SQL report or a Maconomy stand-alone program.

  A Maconomy SQL report is a report that is designed in the Maconomy Report Generator Language (RGL). Such reports are usually executed from a parameter dialog.

These commands and the ability to combine them constitute a powerful tool to integrate Maconomy with other applications.

In the definition of a script, you can use data from the active window or printout as parameters when executing reports or external programs.

This document contains a description of the features in Workflow Spider scripts, followed by some examples and the formal definition of the Maconomy Workflow Spider Scripting Language, the language used for defining the scripts.

Scripts can be initiated in the following ways:

- During the execution of standard printouts.

  Script type: PrintCommand. Scripts of this type can be activated at the end of each Layout Designer page. For an invoice printout, this means at the end of each invoice because a database cursor exists on the page, and the execution of the page is repeated for each invoice that matches the selection criteria. A typical listing printout (for example, a Print Order Listing) does not have a cursor on the page and is only executed once. Consequently, the script is only executed once.

- From the Action menu.

  Script type: ActionMenuItem. You can add server-based scripts to the Action menu in the Java Client; the scripts are also available from Mscript. This type of script does not have access to external programs or files that reside on the client's machine. Thus this type of script corresponds to scripts of the type PrintCommand. It is not possible to return values from an ActionMenuItem script to the dialog.

Note that after the Workflow Spider is installed, it is accessible from all companies in Maconomy. For this reason, it is important that the appropriate access levels and access rights are defined before the Workflow Spider is installed.

Scripting of Maconomy Reports

The principal reason for using scripting of Maconomy reports is the ability to start a report with parameters.

Scripting of Maconomy reports has two purposes:

- Extending standard printouts by including results from Maconomy reports.
- Extending the user interface by adding special “Print This” commands in windows where the standard printouts need customized information. An example of this could be a report that prints a label for the current customer in the Picking List window.

Executing Reports from Printouts

The purpose of scripting in printouts is to extend the possibilities of customizing the output of standard printouts by adding output from Maconomy SQL reports.

Reports can be activated by the standard printout with parameter values transferred from the primary database record (for example, JobHeader in the Print Job Card window) and the variables in the standard print dialogs.

The printed output using Workflow Spider scripting looks like a standard printout with a new block of data.

The reports that are executed by a script are executed after one iteration of a Layout Designer page. This means that the script is executed once for each invoice when printing invoices, but when using the Print Item Card command, for example, the report is only executed once at the end of the print.

The print type could be changed from one application version to another, for example, from iterative to batch printing. However, this is not likely, because a change makes the migration of layout changes from one version to another impossible.

In addition to performing the printout from the report, you can also generate a text file for each execution of the report.

If you only want to create a text file, you can suppress the printout of the report.

Example: Printing a Report for Each Invoice

The InvoiceSpecification.grn report is a report with a parameter dialog that contains one variable, InvoiceNumberVar. After printing an invoice, Maconomy runs the report, using the invoice number from the invoice. The printout from the report starts at the top of the next page.

The name of the printout is the internal name (here P Invoice), because external names (Print Invoice) can be customized and cannot be used as key values. Internal names can be found through the Window Layouts in Maconomy window.

Keywords in the Workflow Spider Scripting Language are emphasized in bold.

```
<Print DialogTitle="P Invoice"
LayoutName="Standard">
<ExecuteReport
ReportFileName="InvoiceSpecification.grn"
InputVariables=(InvoiceNumberVar)
InputValues=(Invoice.InvoiceNumber)
StartingPoint=NextPage
>
<End Print>
```

Example: Exporting Invoice Data during Printing

After adding this script, executing Print Invoice with the layout Standard with Export creates a file for each invoice. For invoice number 1234, the name of the file is EXP1234.TXT.

In the definition of the input values, a single value can be constructed by concatenating database fields and constant strings.

OutFileName is a standard variable in the Workflow Spider Scripting Language. If you set this variable, it contains the name of the file that is to receive the text output from the report. If you do not define the variable, no file is generated.

The text files that are generated from the execution of the report are transferred to the client, unless OutFileName begins with UX. For more information about this, see “Input Variables.”

The parameter DoPrintOut is set to False, which suppresses the printout from the report.

```
<Print DialogTitle="P Invoice"
  LayoutName="Invoice with Export">
  <ExecuteReport
    ReportFileName="ExportReport.grn"
    InputVariables=(OutFileName,InvoiceNu
    mberVar)
    InputValues=("EXP"+Invoice.InvoiceNumber+".TXT",
      Invoice.Invoicenumber)
    DoPrintOut=False
  >
<End Print>
```

Example: Executing Two Reports in One Print

You can execute multiple reports in one print operation by adding instances of the ExecuteReport command, for example, combining the two first examples.

```
<Print DialogTitle="P Invoice"
  LayoutName="Standard with Export">
  <ExecuteReport
    ReportFileName="InvoiceSpecification
    .grn"
    InputVariables=(InvoiceNumberVar)
    InputValues=(Invoice.InvoiceNumber)
    StartingPoint=NextPage
  >
  <ExecuteReport
    ReportFileName="ExportReport.grn"
    InputVariables=(OutFileName,InvoiceNu
    mberVar)
    InputValues=("EXP"+Invoice.InvoiceNumber+".TXT",
      Invoice.Invoicenumber)
    DoPrintOut=False
  >
<End Print>
```

Executing Reports from the Action Menu

You can add reports to the Action menu. The functionality of these reports is the same as described in “Executing Reports from Printouts.”

Example: Printing a Customer Report from Sales Orders

The report CustRep.grn has been developed, showing what a customer has purchased during the last year and what items are currently on backorder.

This report can be added to the Sales Orders window and executed for the customer specified on the current sales order.

```
<ActionMenuItem DialogTitle="SalesOrders"
  MenuTitle="Customer Report">

  <ExecuteReport
    ReportFileName="CustRep.grn"
    InputVariables=(CustomerNumberVar)
    InputValues=(OrderHeader.CustomerNumb
    er)

  >
<End ActionMenuItem>
```

Adding this script to the ActiveScriptDefinition file adds an item to the Special menu in the Sales Orders window. Selecting this menu item executes the report with the current customer number as a parameter.

Example: Executing a Report that Calculates Discount

A customer has a very special calculation of charges that depends on the customer in question. This calculation is not covered by normal Maconomy functionality.

To allow the correct calculation, a special report called CalcCharges.grn was developed. This report takes a customer number as input and returns four charge codes and four amounts.

A script that activates this report and returns the charges to the sales order could be as follows:

```
<ActionMenuItem DialogTitle="SalesOrders"
  MenuTitle="Calculate Charges">

<ExecuteReport
  ReportFileName="CalcCharges.grn"
  InputVariables=(CustomerNumberVar)
  InputValues=(OrderHeader.CustomerN
  umber)

OutputVariables=(

    OrderHeader.ChargeCode1,
    OrderHeader.ChargeCode2,
    OrderHeader.ChargeCode3,
    OrderHeader.ChargeCode4,
    OrderHeader.Charge1Currency,
    OrderHeader.Charge2Currency,
    OrderHeader.Charge3Currency,
    OrderHeader.Charge4Currency)

  OutputValues=(
    ChargeCode1Var,
    ChargeCode2Var,
```

```
                    ChargeCode3Var,
                    ChargeCode4Var,
                    Charge1CurrencyVar,
                    Charge2CurrencyVar,
                    Charge3CurrencyVar,
                    Charge4CurrencyVar)
                >
<End ActionMenuItem>
```

Scripting of External Products

With scripting of external products, Maconomy can launch or activate other applications using parameters from Maconomy, providing an open environment.

Command files (.cmd or .bat files, or shell scripts in UNIX) and executable programs can be started with parameters, as if they were launched from a command line.

External Scripting from Printouts

When executing printouts, you can only launch programs; opening a document on the server makes no sense.

A typical use of external commands on the server could be sending a file (exported by a report) to a customer, using an email or fax application.

Combined with the Automatic Layout Selection feature, introduced in Maconomy 4.0, you can create layouts for each communication method, and Maconomy will be able to send email, fax, or telex depending on the customer.

You do, however, need an external program that can deliver the exported data to the communication system.

One such program could be an email sender (for Windows) called blat, which can be downloaded from the Internet as freeware at http://www.interlog.com/~tcharron/blat.html. Windows does not normally include an email sender.

The name of the executable program is blat.exe, and it is a command-line program. A call to the program could look like the following:

```
blat MessageFileName -tJohnDoe@maconomy.com -s"Hello there"
```

This command sends an email to JohnDoe@maconomy.com, with the subject “Hello there,” with the contents of the message being read from the file named MessageFileName.

This program can, for example, be used in combination with an export from a report to send an email to the customer during the printing of order confirmations. An example of this is appears in the following description.

Since there is no email field in the OrderHeader relation, the **Name5** field is assumed to contain the email address in the following example.

The use of the prefix UX in the specification of OutFileName is described in “Input Variables.”

The subject is surrounded by double quotes in the call to blat, because the subject contains a space. This should be done for all parameter strings that might contain spaces.

Workflow Spider scripts reduce two double quotes in the specification to one, for example, before calling the blat program as shown in the following example.

```
<Print            DialogTitle="Print
OrderConfirmation"
```

```
LayoutName="Standard with E-mail">
<ExecuteReport
  ReportFileName="Exportorder.grn"
  InputVariables=(OutFileName,
  OrderNumberVar) InputValues=(
    "$UX$O"+ OrderHeader.OrderNumber +".txt",
    OrderHeader.OrderNumber)
  DoPrintOut=false
>
<DocumentCommand
  InputParameters=(
    "blat ",
    "D:\MaconomyNT\MaconomyHomes\w_7_0\MaconomyOutput\"+
    "O"+OrderHeader.OrderNumber+".txt ",
    "-t"+OrderHeader.Name5+" ",
    "-s""Order Confirmation""")
>
<End Print>
```

External Scripting from the Action Menu

You can add reports to the Action menu. The functionality of these reports is the same as described in “External Scripting from Printouts.”

Grammar of the Workflow Spider Scripting Language

The `ActiveScriptDefinition` file (see “Configuration”) can contain comments. A comment starts with “--” (two dashes) on a line, and the rest of the line is interpreted as the comment.

The language is not case-sensitive, but names of external commands or documents on UNIX platforms are case-sensitive.

The syntax for the Workflow Spider Scripting Language is described by the following grammar in Extended Backus Naur Form (BNF).

Terminal symbols are printed in a monospace font; non-terminal symbols are printed in *lower case italic letters*, and grammar symbols in *UPPER CASE ITALIC LETTERS*.

The BNF constructs that are used are the following:

- r s – meaning r followed by s.
- r | s – meaning r or s.
- (r)* – meaning zero or more r.
- (r)+ – meaning 1 or more r.

The grammar rules are outlined in the following table.

Grammar Rule	Elements in Rule
start	<ActiveScripts> (Script)+ <End ActiveScripts>
Script	MenuItem | PrintCommand | ActionMenuCommand

Grammar Rule	Elements in Rule
ActionMenuItem	<ActionMenuItem (MenuItemAttribute)*> (Command)+ <End ActionMenuItem>
PrintCommand	<Print (PrintCommandAttribute)*> (Command)+ <End Print>
MenuItemAttribute	DialogTitle | MenuTitle
PrintCommandAttribute	DialogTitle | LayoutName | Wildcard
DialogTitle	DialogTitle = STRING
MenuTitle	MenuTitle = STRING
Wildcard	“”
LayoutName	LayoutName = STRING
Command	ExecuteReport | DocumentCommand | PerformEnter | RedrawWindow
ExecuteReport	<ExecuteReport (ReportParameter)*>
DocumentCommand	<DocumentCommand (DocumentParameter)*>
PerformEnter	<PerformEnter>
RedrawWindow	<RedrawWindow>
ReportParameter	ReportFileName | DoPrintOut | StartingPoint | InputVariables | InputValues
DocumentParameter	InputParameters
ReportFileName	ReportFileName = STRING
DoPrintOut	DoPrintOut = BOOLEAN
StartingPoint	StartingPoint = StartingPointTypes
InputVariables	InputVariables = (VariableList)
InputValues	InputValues = (ValueList)
OutputVariables	OutputVariables = (OutputVariableList)
OutputValues	OutputValues = (OutputValueList)
InputParameters	InputParameters = (ValueList)
StartingPointTypes	Top | Below | NextPage
VariableList	Variable (,Variable)+
ValueList	Value (,Value)*

Grammar Rule	Elements in Rule
OutputVariableList	VariableOrDatabaseField (,VariableOrDatabaseField)+
VariableOrDatabaseField	Variable | DatabaseField
Variable	ID
Value	ValueItem (+ ValueItem)*
ValueItem	DatabaseField| Variable | STRING
DatabaseField	RelationName.FieldName
RelationName	ID
FieldName	ID

The meaning of the grammar symbols used in the grammar is:

- BOOLEAN – Either false or true.
- STRING – A text in quotes, such as "`AccountCard`."
- ID – An identifier, such as the name of a database field or a variable, such as `AccountText`.

Description of Commands

This section contains a description of all commands in the workflow Spider Language.

- **ExecuteReport** — This command is used for executing Maconomy SQL reports (RGL reports, generated by the Maconomy Report Generator) and Maconomy standard import programs. The file name extension of these files is usually .grn for reports and .prm for import programs.
- **DocumentCommand** — This command is used when you want to execute an external program.
- **PerformEnter** — This command can be used to simulate a user pressing Enter after the script has been executed. This can be useful, for example, if the script returns output values from a report, because Enter must be pressed for the data to be updated in Maconomy's database. If you do not use a PerformEnter command, the user must press Enter manually to confirm the changes in the window.
- **RedrawWindow** — In Workflow Spider you can update the contents of the current window by running an import program that modifies the entry currently shown in the window. When doing this, the client program is not aware that the current data that is being shown is outdated, meaning that the window is not automatically updated with the new information. To update the window so that it shows the new data, you can use the RedrawWindow command, which causes the entry currently shown in the window to be refreshed with the updated information from the database.

Description of Attributes

This section contains a description of all attributes in the Workflow Spider Scripting Language.

- **DialogTitle** — When defining a MenuItem, the `DialogTitle` defines the dialog (data entry window such as Sales Orders) in which the Special menu item should be added. If `DialogTitle` is an empty string, the `MenuItem` is defined in all windows.

Menu items defined for search windows and parameter windows have no effect, but general menu items are effective in such windows. You can, for example, execute a script that opens a Word document from a search window.

The possible value of `DialogTitle` is the internal name of a dialog, which is also used in the Layout Designer. The reason for this is that the current window title that the user sees can be changed and cannot be used as a unique key.

In the search window in the Window Layouts window in the Set-Up module, you can see all internal and standard external window titles. For example, the internal name for Print Invoice is `P Invoice`.

- **MenuTitle** — The `MenuTitle` command defines the title for the `MenuItem` script in the Special menu.
- **LayoutName** — The `LayoutName` command defines the name of the layout, which is shown in the print dialogs. If `LayoutName` is an empty string, the script is defined for all existing layouts of the print. Otherwise, the script is only run when the specified print layout is used.
- **DoPrintOut** — If this attribute is set to `False`, the execution of the report produces no printout. This can be used if you only want to generate a text file. The default value is `True`.
- **InputParameters** — The `InputParameters` command defines a document or a system command to be executed.

  Input parameters are constructed from constant strings and/or from values retrieved from the current dialog or printout.

  When opening a document, only one parameter is needed: the name of the document.

  When opening a program with parameters, the first input parameter must be the program name, and the parameters to the program must be input parameters `2..n`.

  A single input parameter can be constructed by concatenating constant strings and the values of fields from the current dialog or printout. Values of field and variables of types other than strings are converted to strings, in the same way that they are converted when shown on the screen.

  For example, a filename can be constructed from constant strings and the job number:

  ```
  InputParameters = ("TestProg2","FILE"+JobHeader.JobNumber+".TXT")
  ```

  The resulting program call would look like this:

  ```
  TestProg2 FILE250001.TXT
  ```

  if the job number was 250001.

  For both `InputParameters` and `InputVariables`, a few “pseudo variables” exist. These variables are defined by the system and can be used as any other variable. The pseudo variables are:

  - `UserName` — The name of the current user.
  - `ShortName` — The identification of the current database. This is the short name entered for the database during installation.
- **InputVariables** — `InputVariables` are the variables in the report’s selection criteria, to be initialized from `InputValues` before calling the Maconomy report.

For all reports, there is a special variable called `OutFileName`. A text file is only generated if this variable is part of `InputVariables`. The value of the variable is used as the filename.

If the filename has the prefix UX, the file is located on the server; otherwise, it is transferred to the client after the execution of all scripts in the `MenuItem` script or the completion of the printout.

If multiple reports are executed in a row, and the same `OutFileName` is used, the file is overwritten for each execution.

When using UX to create files on the server, the filename can either be an absolute path (starting with / on UNIX or a drive letter on NT), or a relative path name, such as `FILE250001.TXT`.

If the filename is a relative path, the file is placed relative to the folder `MaconomyOutput` in the `MaconomyHome` folder.

When opening a program with parameters, the first input parameter must be the name (including the path if necessary) of the program, and the parameters to the program must be input parameters `2..n`.

If parameters include spaces or other special characters, use quotes around the parameters to prevent the command shell from interpreting these characters.

Example:

```
"/data/Maconomy/us4_0/MaconomyOutput/FILE 250001.TXT"
```

or

```
"d:\MaconomyNT\MaconomyHomes\us4_0\MaconomyOutput\FILE 250001.TXT"
```

A few “pseudo variables” defined by the system exist. For more information, see “Input Parameters.”

- **Prompting for a value** — Sometimes when invoking a Workflow Spider script from a dialog, you cannot retrieve all of the parameter values from the dialog. To solve this problem, you can activate a dialog box that prompts for a value. For instance, consider a report that requires a birthday as a parameter to find the employees who were born before a given date.

  Executing a script with the line displays a dialog box.

  ```
  InputParameters= ("prompt:Please Enter a birthday")
  ```

  When the user clicks the OK button, the entered value replaces the `Prompt:...,` in the parameter specification, thus acting as input values for the relevant input parameter. If the user clicks the Cancel button, the entire Workflow Spider script is canceled.

- **InputValues** — When executing Maconomy reports, the values assigned to the variables in the selection criteria are defined by the `InputValues`.

  A value must be specified for each variable in the input variables. The values are defined in the same way as `InputParameters`.

- **OutputVariables** — When executing reports from the Special menu, the attribute `OutputVariables` specifies the database fields and variables in the active window to which the output values from the report execution should be written.

  It is only possible to return values to open fields in the active window. Closed fields in the window are not updated. This means that only data that could have been changed manually by the user can be changed through Workflow Spider.

- **OutputValues** — When executing reports from the Special menu, the attribute `OutputValues` specifies the values to be returned to the active window.

  Output values can be specified in the same way as `InputParameters`. The only difference is the fact that the variables that are specified are variables that are defined in the report, and that database fields cannot be specified.

- **ReportFileName** — This command defines the name of the file that contains the report to be executed. The Maconomy Report Designer for Windows must compile the report.

  Maconomy looks for the report on the server, in the directory ActiveScripts in the MaconomyDir directory, for example:

  ```
  /data/maconomy/w_15_0/MaconomyDir/ActiveScripts
  ```

- **StartingPoint** — The `StartingPoint` attribute can be used to control where to start the printout of the Maconomy report when it is executed during a standard printout.

  There are three options:

  - Top — The printout of the report starts at the top of the last page. When using this option, the developer can position the printout from the report exactly on the page, for example, using the report to print a special footer.
  - Below — The printout of the report is started just below the last item, on the last page. This is the default option.
  - NextPage — The system performs a page eject before printing the report.

System Requirements and Configuration

This section describes the requirements for the Maconomy Workflow Spider, and how and where to configure it.

In addition, this section includes some notes about upgrading Workflow Spider scripts from one application version to another.

System requirements

Before you configure Active Scripting, one or both of the following add-on products must be installed:

- Add-on 54: Active Scripting, Maconomy Products
- Add-on 55: Active Scripting, External Products

Configuration

Maconomy reports that are referenced in scripts must be stored on the server in a folder named `ActiveScripts` in the `Maconomy` folder, for example:

```
D:\MaconomyNT\MaconomyHomes\w_17_0\MaconomyDir\ActiveScripts
```

Scripts are defined in a file called `ActiveScriptDefinition` in the `Definitions` folder, for example

```
D:\MaconomyNT\MaconomyHomes\w_17_0\MaconomyDir\Definitions\ActiveScriptDefinition
```

When starting a client, the client and the server read the file.

Syntax errors are only reported on the client. Semantic errors, for example, references to nonexistent fields or variables in scripts, are not reported until the scripts are executed.

Scripts are active for all users, and cannot be modified by ordinary users, only by system administrators.

ActiveScriptDefinition

The `ActiveScriptDefinition` file contains all of the scripts that are available to the clients. “Grammar of the Workflow Spider Scripting Language” describes the formal syntax of the file.

The following is an annotated example of an active script definition file.

```
1  < ActiveScripts >
2  < ActionMenuItem
                    DialogTitle="Seasona
   lmodels" MenuTitle="Export/import
   Line" >

   <ExecuteReport
     ReportFileName="expseasonalmodel.grn"
     DoPrintOut=false

     InputVariables=(OutFileName , SmodelVar, LinenumberVar)
     InputValues=("$UX$S"+SeasonalModelline.SeasonalModelnumber.
     txt",

       SeasonalModelline.SeasonalModelnumber,
       SeasonalModelline.linenumber) >

   <ExecuteReport ReportFileName="Import Seasonal Models
     4.1.prm" InputVariables=(InFileNameVar, OutFileNameVar)
     InputValues=("E:\MaconomyNT\MaconomyHomes\W_7_0"+

       "\MaconomyOutput\S"+SeasonalModelline.SeasonalModelnumber
       .txt", "ImportErrors.txt") >

   <end ActionMenuItem >

3  <end ActiveScripts >
```

Annotations

1. Indicates the beginning of the definition of Workflow Spider scripts.
2. This script appears in the Action menu of the Seasonal Models window. This means that it is also available to the Maconomy client for the Java™ platform. No client-side programs can be run from this script.
3. This ends the definition of Workflow Spider scripts for this Maconomy server.

Upgrading

When upgrading a Maconomy installation from one application version to another, all reports used in the Workflow Spider scripts must be recompiled to the new application and installed in the `ActiveScripts` folder in the `Maconomy` folder for the new version.

The `ActiveScriptDefinition` file can normally just be moved from one application version to another. However, if database fields or variables have been renamed or removed, the file must be changed if the changed fields are referenced in scripts.

MCron Overview

MCron is a tool that enables you to execute commands automatically at specified times or dates. You create a configuration file that contains a description of the commands to run and when to run them. MCron then takes care of executing the commands and reporting errors.

MCron is implemented for Windows and Unix. On Windows it runs as a service, and on Unix it runs as a daemon.

The combination of a command and the time(s) on which it should be run is called a job.

You can, for example, use MCron to set up automatic cleanup of log files, running reports at off hours, or activating asynchronous workflow monitors.

Installation

MCron is installed by MConfig as part of a TPU installation. MCron is included in TPU versions 9.0p4 and later. MConfig version 5.6 or later must be used.

When you install a TPU that includes MCron, MConfig creates a default version of MCronConfig if the file does not already exist. MCronConfig is never modified by MConfig. In this way, modifications to MCronConfig survive upgrades of the Maconomy system.

Configuration

The configuration of MCron is done in three places:

- The file MCronConfig. This file contains settings for MCron (for example controlling the amount of log information that is produced), and optionally a specification of jobs for MCron.
- Files from the installed Maconomy applications. The next section describes which files are used. These files only contain MCron job specifications.
- Ini files that determine which of the installed Maconomy applications MCron should consider when looking for job specification files.

Terminology

In the following, the term “application-specific” is used to describe a configuration file for MCron that is part of an installed Maconomy application. The term “configuration file” is used to describe either the file MCronConfig or one of the application-specific configuration files.

The term “ini file” refers to a file used by MCron to determine the Maconomy applications to which it should apply.

Ini Files

When MCron is running, it monitors the IniFiles directory for files named:

```
MCron.<applicationname>.<shortname>.I
```

Each of these files contains a line of the form:

```
APPLICATIONHOME=<path>
```

The ini files tell MCron which Maconomy applications to consider when looking for job specifications, and which shortnames are available (the latter is relevant when using the <<Shortname>> placeholder as described in “Placeholders.”

MCron only executes jobs for a Maconomy application if an ini file exists with APPLICATIONHOME pointing to the application.

The ini files are maintained by the MConfig application (see the MConfig User Manual).

Application-Specific Configuration Files

When MCron is running, it looks for installed Maconomy applications, and for every application MCron uses the following files:

- All files in the MaconomyDir/MCron directory
- All files in the CustomizationDir/Solution/MCron directory if the directory exists
- All files in the CustomizationDir/Custom/MCron directory if the directory exists

These three directories contain the Standard, Solution, and Custom layers for the Maconomy application. The order of these levels is: Standard, Solution, and Custom, where Custom is the topmost layer.

Stamped Configuration Files

MCron only uses application-specific configuration files on the Standard and Solution layers if they are stamped by Maconomy R&D using the MStamper tool. Consequently, consultants and customers can only place new files in the Custom layer.

Syntax of a Configuration File

The following provides a description of the structure of a configuration file. Each line in the MCronConfig file should:

- contain a job specification, or
- contain an MCron setting, or
- start with "–" to mark the line as a comment, or
- contain only whitespace

An application-specific configuration file has the same structure as the file MCronConfig, except that MCron settings are not allowed in those files.

The next section describes job specifications.

Job Specifications

A job specification is a line in a configuration file that contains 7 tokens:

```
jobid  minute hour   dayOfMonth    month  dayOfWeek     command
```

The tokens can be separated by whitespace (spaces and tabs). Note that a job specification must end with a newline character.

The token jobid is a name for the job. The next five tokens describe when the job should be run. The token command contains the command line that is to be executed. The tokens have the following meanings.

Token	Example	Description
jobid	myjob1	This is an identifier for the job. If the job specification is placed in MCronConfig, the jobid must be unique within that file. Otherwise, the job specification is inside one of the application-specific configuration files, and then the jobid must be unique across all application-specific configuration files for the application. Note that jobid values are case-insensitive, and that the character " " is illegal in jobids.
minute	00,30	This controls at which minutes the command should be executed. It is a list of numbers between "00" and "59."
hour	00,07,17,22	This controls at which hours the command should be executed. It is a list of numbers between "00" and "23."
dayOfMonth	01	This controls on which dates the command should be executed. It is a list of numbers between "01" and "31."
month	01, 04, 07, 10	This controls in which months the command should be executed. It is a list of numbers between "01" (January) and "12" (December).
dayOfWeek	0, 3, 5	This controls on which weekdays the command should be executed. It is a list of numbers between "0" (Sunday) and "6" (Saturday).
command	C:\MCronJobs\my job.1 arg1 arg2	This is the command to execute It is passed to a shell for execution. Everything from the start of the command token to the newline character (not included) is considered to be part of the command.

Comma-separated lists cannot contain whitespace.

Placeholders

When creating job specifications in application-specific configuration files, a number of placeholders can be used: <<ApplicationHome>>, <<ApplicationName>>, <<GlobalBinDir>>, <<BinDir>>, <<MaconomyServer>>, and <<Shortname>>. MCron expands these at run time. The placeholders have the following meanings.

Placeholder	Description
<<ApplicationHome>>	Replaced by the path to the Maconomy application that contains the job specification.
<<ApplicationName>>	Replaced by the name of the Maconomy application that contains the job specification.
<<GlobalBinDir>>	Replaced by the path to the directory that contains the global tools, for example, c:\maconomy\bin on Windows.
<<BinDir>>	Replaced by the path to the directory that contains the tools that the application uses.
<<MaconomyServer>>	Replaced by the name of the MaconomyServer executable used by the application that contains the job specification.
<<Shortname>>	This signifies that the job specification is duplicated once for every shortname that MCron knows about for the Maconomy application that contains the job specification (see more in the following).

The placeholder <<Shortname>> is different from the other placeholders in that it means that the job specification is duplicated once for every shortname that MCron knows about for the Maconomy application that contains the job specification. This means that the job is executed for "all shortnames." For example, if MCron is activated for the shortnames w90a and w90b the job specification

```
myjob 00 00 01 01 0 echo "<<Shortname>>"
```

is expanded to

```
w90a.myjob 00 00 01 01 0 echo "w90a"
w90b.myjob 00 00 01 01 0 echo "w90b"
```

That is, the <<Shortname>> placeholder is replaced by the values of the known shortnames, and the shortname is prepended to the jobid.

In MCronConfig only the placeholder <<GlobalBinDir>> is allowed.

Shortcuts

MCron allows three kinds of shortcuts that can be used with the tokens minute, hour, dayOfMonth, month, and dayOfWeek to ease specification:

- **Ranges** — A list element can be a range of the form n1-n2. For example, 3-6 means 3,4,5,6.
- **Step values** — A range can be followed by a step value: n1-n2/n3. For example 1- 13/2 means 1,3,5,7,9,11,13.
- **Star** — A list element can be an asterisk *. This means "all allowed values for the given token." For example, an asterisk in the hour token means the same as 00-23.

The following table describes the meaning of * for every token.

The shortcuts can be combined. For example, 1,4-10/2,23 means 1, 4, 6, 8., 10, 23, and */10 in the minute token means 00,10,20,30,40,50.

Token	Means
minute	00-59
hour	00-23
dayOfMonth	01-21
month	01-12
dayOfWeek	0-6

Semantics

A job is executed whenever the following conditions are met:

- The current minute is in the list described by minute.
- The current hour is in the list described by hour.
- The current date is in the list described by dayOfMonth.
- The current month is in the list described by month.
- The current weekday is in the list described by dayOfWeek.

This means that for the job specification

```
--logname  min   hour  dayOfMonth month  dayOfWeek  command
  myjob    00    15    01,15       *      1          c:\mycmd
```

the command is executed at 15:00 every Monday that falls on the first or fifteenth of the month.

This behavior can be described as

```
minute && hour && dayOfMonth && month && dayOfWeek
```

Note that the standard cron utility on Unix uses a different behavior:

```
minute && hour && month && (dayOfMonth || dayOfWeek)
```

Examples

Run cmd at 14:00 on the first of every month:

```
--id min      hour   dom    mon    dow
       cmd j1 00     14     1      *
       *      cmd
```

Run cmd every fifteen minutes:

```
j2     */15 * *      *      *      cmd
```

Run cmd every minute:

```
j3     *      *      *      *      *      cmd
```

Run cmd at the start of every hour:

```
j4     00     *      *      *      *      cmd
```

Run cmd every third hour:

```
j5      00      */3     *       *       *       cmd
```

Run cmd every midday between the first and the fifteenth, as well as the 20th and the 25th (inclusive) and also on the 17th of every month:

```
j6 00 12      1-15,17,20-25 * *     cmd
```

MCron Settings

As mentioned, the file MCronConfig contains some settings for MCron. The following table describes these settings.

Setting	Mandatory	Values	Default Value	Description
LOG	No	none startstop debug	None	Determines how much log information MCron should produce.
LOGFILE	No	A filename	The file MCronLog.cvs in the tmp directory used by Maconomy	The name of the log file to use (only relevant if LOG is set to none).
SYSTEMLOG	No	none startstop debug	None	Determines how much log information MCron should write to the Event Log in Windows. This setting is ignored in Unix.
SHELL	No	A filename	/bin/sh	Determines which shell to use for executing the commands in Unix. This setting is ignored in Windows.

The settings are specified using the following syntax:

```
setting = value
```

Note that there must be whitespace on either side of the equals sign.

The values for LOG and SYSTEMLOG are described in “Logging.”

How MCron Works

When the MCron daemon or service starts, it reads the configuration and enters its “main loop.” This is continued until MCron is stopped by the user. The “main loop” is performed once every minute. The loop consists of the following tasks:

- MCron looks for ini files. If new ones have been added or existing ones deleted, MCron re-reads its entire configuration. If one of the ini files has been modified since the last check, MCron re-reads the jobs for the corresponding Maconomy application.
- MCron checks all Maconomy applications that it knows about to see if files have been added or removed from any of the MCron directories (see “Ini Files”). If this is the case, MCron discards all information about jobs for the application, and reads the configuration files using the rules described previously.

- If at least one of the configuration files has changed for a Maconomy application since the last loop, all job specification files for that application are re-read.
- MCron builds a list of the jobs to be executed in the current minute.
- MCron starts each job on the list in a separate process.

Error Reporting

When a job terminates, MCron tries to determine if the job was successful or not.[3] If a job failed, the output from the command (that is, what was written to standard-out and standard-error) is appended to the log file for the job.

If MCron encounters a “Critical error”, a message is written to PPU_DebugStr. A critical error might, for example, be:

- A syntax error in one of the configuration files.
- MCron could not allocate memory for running a job.

Logging

If LOG is set to none in the file MCronConfig, no additional information is logged. If LOG is set to startstop, MCron writes a message to the log file (determined by the LOGFILE setting) when executing a job and when the job terminates. If LOG is set to debug, additional debugging information is written to the log file.

The value for SYSTEMLOG is used in the same way as the LOG setting; only it controls what should be written to the Windows Event Log. Be aware that with the SYSTEMLOG setting set to debug, a lot of information is written to the Event Log, which could then fill up or overwrite information written by other programs.

Log Format

The file format of the log file is semicolon-separated values. This format can be read by MS Excel and can easily be parsed by, for example, an awk program.

A log message consists of a single line in the log file. Each log message has the following format:

```
year;month;date;hour;minute;second;logtype;jobid;msg
```

The first five fields tell when the log message was generated. The field logtype can be either debug or startstop, depending on the type of information contained in the field msg. If the logtype is startstop, the field jobid contains the name of the job for which the msg was generated. If the logtype is none, the field jobid is empty.

Interacting with the Computer Clock

MCron uses the clock on the computer it runs on. All times and dates in the job specifications are interpreted as local times on this machine.

Also, MCron depends on the system clock for making sure that it checks and runs jobs once every minute. If the computer is busy, or the system clock is unstable, MCron may not be able to do this exactly once every minute. For this reason, MCron keeps track of the time since it was last “awake.” If it wakes up more than one minute after its last run time, it starts all jobs scheduled to run at least once in the period between the last run time and the current time. Note that even if a job is scheduled to run more than once in this period, it is only started once.

[3] MCron assumes that a return value of 0 means that no errors occurred

If the time has been adjusted backward, meaning that the current time is earlier than the last run time, MCron waits until time has “caught up” before any jobs are run.

For example, when daylight saving time (DST) starts, the time goes from 01:59:59 to 03:00:00. When MCron wakes up, it starts all jobs that are scheduled to run at least once in the interval [02:00,03:00].

When DST ends, the time goes from 02:59:59 to 02:00:00. MCron does not start any jobs until the time is 03:00. In this way jobs scheduled to run between 02:00 and 03:00 are not repeated when DST ends.

Running MCron

This section describes how to start and stop MCron. The command-line options for MCron are summarized in the following table.

Option	Description
-h	Display help information.
-v	Display version information
-vv	Display extended version information.
-i <user> <password> or -install <user> <password>	Windows only. Install MCron as a Windows service. The service should run as the specified user.
-r or -remove	Windows only. Remove the Windows service MCron.
-s [HH:MM] or -start [HH:MM]	Windows only. Start the Windows service MCron. The optional argument specifies which “last run time” to assume. Using this option has the same effect as pressing “Start” in the “Services” part of the Microsoft Management Console.
-stop	Windows only. Stop the Windows service MCron. Using this option has the same effect as pressing “Stop” in the “Services” part of the Microsoft Management Console.
-f [HH:MM] or -foreground [H:MM]	Run MCron in “foreground mode.” The optional argument specifies which “last run time” to assume.
HH:MM	Unix only. Start the MCron daemon, and use the specified time as “last run time.”

Foreground Mode

MCron can run in two modes: foreground or service/daemon. In foreground mode, all logging that is written to the log file is also written to the console. When MCron runs in foreground mode, pressing CTRL+C will terminate MCron.

Last Run Time

As described previously, MCron keeps track of the last time that it checked and ran jobs. When MCron is launched, it sets the “last run time” to the previous minute, but by specifying a time on the command line this value can be overwritten.

Running MCron on Windows

When MCron has been installed as a Windows service, it can be started by using the command line option -start or by using the “Services” part of the Microsoft Management Console. When MCron is installed with the -install option, it starts automatically when Windows is started.

MCron can be stopped by using the -stop command line option, or by pressing “Stop” in the Microsoft Management Console.

Running MCron on Unix

On Unix, the MCron daemon is started by issuing the command MCron. To prevent multiple instances of MCron running, a pid file is used. When MCron is launched, it attempts to get an exclusive lock on this file, and if successful, it writes its Process ID (pid) to the file. If an exclusive lock could not be obtained because MCron is already running, the new instance terminates.

MCron can be stopped by sending the SIGTERM signal to MCron. Here the number in the pid file is useful: to send the SIGTERM signal to MCron, issue the following command (where “>” is the shell prompt):

```
> kill -s SIGTERM 'cat /usr/maconomy/MCron.pid'
```

To force MCron to reread its configuration next time it is “awake”, send the signal

SIGHUP:

```
> kill -s SIGHUP 'cat /usr/maconomy/MCron.pid'
```

When SIGHUP is received, MCron also closes and reopens the log file. This is useful if you want to create a log file (for example, if using log rotation).

What Happens when MCron Stops

Previous sections described how to stop MCron. When MCron is stopped, only the “main program” stops; any running jobs keep running. Note, however, that on Windows the return status of these jobs is not checked. This means that the log will not contain entries about their status, and the job logs will not be updated in case of errors.

Cleaning up after MCron

MCron creates temporary files for storing output from jobs. If a job terminates with an error, this information is copied to the job log file. Normally the temporary files are deleted, but there are situations where they are left on disk:

- On Windows: If MCron is stopped while jobs are running, the temporary files of the running jobs are left intact, because they could be the only way to get the status of the job.
- On Unix: If the process that monitors a job is killed, the temporary file will not be deleted.

File Locations

The following table contains an overview of the files that are used by MCron.

File	Operation	Location	Unix Attributes
MCronConfig	Read	On Windows: c:\maconomy\IniFiles[4] On Unix: /usr/Maconomy	-rw--------r (0600)
MCron.<applicationname>.<shortname>.l	Read	On Windows: c:\maconomy\IniFiles[2] On Unix: /usr/maconomy	-rw--------r (0600)
“log file”	Write	Determined by the LOG setting in MCronConfig	-rw--------r (0600)
“job log file”	Write	On Windows: c:\maconomy\tmp\[2] On Unix: /tmp	-rw--------r (0600)
Mcron.pid[5]	Read/Write	/usr/Maconomy	-rw-r—r—r (0644)
“temporary files”	Write	On Windows: c:Maconomy\tmp\MCxxxxx.log[2] On Unix: /tmp/MCxxxxx.log	-rw--------r (0600)

The column “Unix attributes” describes the file access permissions used on Unix systems. They are set so that only the Maconomy user has access to the files. One exception is the pid file, which is readable by everyone.. If the log file exists when MCron starts, the access permissions of the existing file are left untouched.

As described previously, each job uses a separate log file for error reporting. For jobs from MCronConfig, the file name is MC_Global_JOBID.log, where JOBID is the job’s ID. For jobs from application-specific configuration files, the filename is MC_APPLICATIONNAME_JOBID.log, where APPLICATIONNAME is the name of the application and JOBID is the job’s ID. As described previously, if the placeholder <<Shortname>> is used, the shortname is prepended to the job ID from the input file. This means that the name of the log file will be MC_APPLICATIONNAME_SHORTNAME.JOBID.log, where APPLICATIONNAME is the name of the application, JOBID is the job’s ID, and SHORTNAME is the shortname.

Implementation Notes

Windows

On Windows, MCron runs in a single process. The main thread performs the “main loop” described previously. To start a job, the main thread creates a thread T. The thread T creates a process P that executes the command for the job. T waits for P to terminate. When P terminates, T examines the return value and handles error reporting before terminating.

[4] The drive letter depends on where the Maconomy system is installed.
[5] Unix only

The Windows version of MCron uses WaitForSingleObject and WaitForMultipleObjects to sleep and to wait for a thread to terminate, or a Stop event from the user.

Unix

On Unix, the daemon forks and terminates the parent, as is standard behavior for a daemon. The child process P performs the “main loop” (as described previously). To start a new job, the process P forks and continues in the main loop. The child process C forks again. This time the child process C2 executes the command (with exec) and terminates. The process C waits for C2 to terminate, and examines the return value before it terminates.

The main process in the Unix version of MCron catches three signals: SIGTERM, SIGHUP, and SIGCHLD. The signal SIGCHLD is sent to the main process whenever one of its child processes terminates. To avoid “zombie” processes, the main process uses waitpid to free system resources uses by the child.

File locking on the pid file is implemented by using fcntl, which is perhaps implemented by using alarm, so MCron should not be modified to catch SIGALARM.

Tips

- By looking at the timestamp for “last modification” for the job log files, it can be determined when the last error from a specific job occurred (if any).
- If a configuration file is changed while MCron is running, and the modified file contains errors, MCron will continue with the previous configuration. This is reported in PPU_DebugStr. It is therefore a good idea to monitor PPU_DebugStr a few minutes after saving changes to the configuration file if MCron is running.
- Changing the system clock affects MCron as described previously. For example, if time was adjusted backwards, MCron will not run any jobs until time has “caught up.” To force MCron to run jobs as if the time had not been reset, stop and restart MCron (possibly with a value for “last run time”). This works because “last run time” is not saved when MCron terminates.
- If the system clock has been adjusted backwards, and the MCron configuration is changed, the file modification time is not later than the old configuration, so MCron will not re-read it. On Unix you can send the SIGHUP signal to force MCron to re-read its configuration.
- The amount of logging can be changed while MCron is running by changing the configuration file.
- A job can be (temporarily) disabled by inserting the comment marker (--) in front of the line in the configuration file while MCron is running. Note that this only works for configuration files in the Custom layer, because configuration files in the Standard and Solution layers must be stamped, and hence cannot be modified.
- If you want to disable a job specified in the file f in either the Standard or Solution layers, you can create an empty file named f in the Custom layer. This will in effect disable all jobs described in the original file f. If the original f contains more than one job specification, and you only want to disable some of them, copy the job specifications of the other jobs to the file f in the Custom layer.
- MCron does not check whether a command has ended before starting another instance.

  For example, if a job takes two minutes to complete, but it is started in two consecutive minutes, two instances of the same command will run. If having multiple running instances of a job is a problem, the logic to enforce it must be implemented in the job’s command.

Getting Started with Localization

Before You Begin

Before you begin, complete the following:

- Prepare for the translation by reading the latest documentation. Know the terminology used in the existing dictionary and the changes you plan to add. Knowing the functionality in which the strings to be localized are used are key factors for quality translations.
- Learn the special notations used in the Maconomy dictionaries; see the “One Page Dictionary Guide.”
- Install a good text editor. The preferred application in which to work with the dictionary is a regular text editor preferably with good search capabilities. Examples include: Crimson and Notepad++.

“Smart” programs like Microsoft Word or Excel are not recommended as they will often silently change the text, such as the number of spaces, or "quotes" to “quotes”.

Try out the helping tools provided by Deltek. They can save you a lot of time and trouble, and will also result in higher quality. For further information about the tools and resources that can help you in your work, see “Common Issues to Avoid.”

Whether you translate manually, or use a Custom Dictionary Builder, following are some pitfalls to avoid:

- Term variations: A term may appear in several forms e.g. “Project manager” appears also as “Project Mgr”, “Project Man.”, “Proj. Manager”, “ProjectManager”, or even just “PM”.
- Case variations: “Job name”, “Job Name”, “job name” should be considered individually. English have a tradition for uppercasing all words in field labels, but other languages have different conventions.
- Correct grammar: Changing Job to Engagement implies changing “**a** job” to “**an** engagement”. In other languages a change in the grammatical gender of words will have a similar effect on other neighbouring words. Also consider plurals like “Proj. Mgr**s**.”.
- The string length should be taken into consideration. Most of the Maconomy system can only handle strings up to 256 characters. In addition, layouts (both print and screen layouts) are to some degree sensitive to changing the length of field titles. Maconomy will try to adjust the layouts, but longer strings may look odd or cause printing beyond the paper margin.

Become Familiar with Documentation

The first thing you need to do before starting work on a missing list is to read the documentation on the procedures, guidelines and requirements to formats and notation. Currently, this documentation is available:

- **Dictionary Format and Notation manual**
  The strings in the missing list originate from different sources in the Maconomy software, not just windows and printouts. Error messages, menu titles, and tool tips are also part of the dictionary, in addition to import programs and reports. You will therefore encounter placeholders, special characters, and notations that need to be handled in a specific way. This manual describes the most common types of notations and line types, thus making it easier for you to achieve the correct notation in the translation. The manual also describes how genders are specified and how to manage gender-dependent grammar in the dictionary.

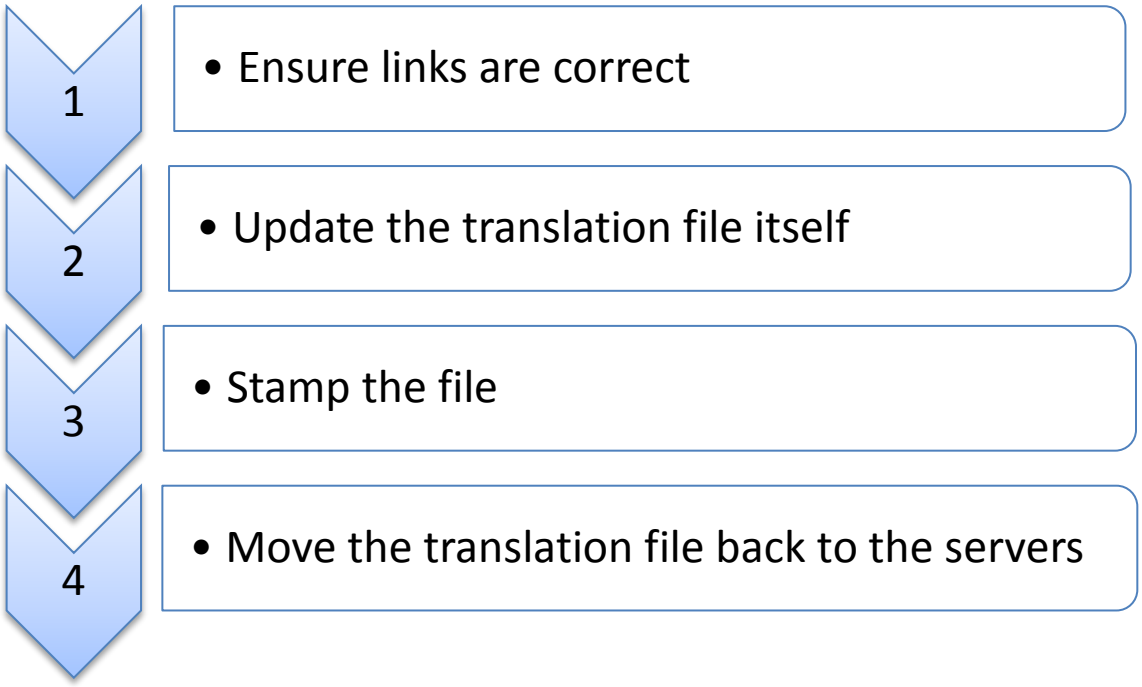
- **Maconomy reference manual/online help**
  The reference manual/online help provides detailed information about the functionality of the Maconomy system and the Portal components. Make sure you have it.
- **Release Notes / Enhancements Guide**
  The Release Notes and Enhancements Guide describe the new features introduced from one version of Maconomy to the next. By reading this information for the version whose missing list you are translating, you are acquainted with the functionality behind the strings in the missing list, which is a shortcut to deciding on the correct terminology.

How to Update a Translation File

This section provides instructions on how to complete each of the steps to update translation files (also known as Dictionaries or Portal Localization). It does not cover what happens if there are errors in the files or if anything in general should go wrong. Therefore it is advised to have a technical consultant on call during this process.

Workflow

This process of updating translation files is divided into four main steps, described below.



"How to" Section

Ensure Links Are Correct

Maconomy installations come with Standard and Solution translation files. When Maconomy is installed, make a new file with the terms that need a different translation for the client, and thus it becomes a "client"-specific translation file.

Always make changes to the client-specific translation files to ensure that changes are not overwritten during upgrades or new service packs.

Example of a Standard translation file:

Translations.W_17_0.en_US

Examples of Solution translation files:

Translations.en_US_CPA

Translations.sv_SE_MCS

where

- Lowercase letters specify language
- First uppercase letters specify country / region
- Last uppercase letters specify dialect, Maconomy solution, or customer specification

Example of a client-specific translation file:

Translations. W_17_0.fr_FR_ORC or W_17_0.sv_SE_MCSOCR

To make sure that links are correct, complete the following step:

1. Confirm that the links you have "point" to the client-specific translation files and not the original. For example, filenames that include either MCS or MAS are original.

   Examples of links pointed to Solution files delivered with Maconomy:

   Java: http://serp02.orckestra.local:83/cgi-bin/Maconomy/Jaconomy.orctrain.US_MCS.exe/Maconomy/JaconomyMP/webstart.jnlp

   Portal: http://serp02.orckestra.local:83/cgi-bin/Maconomy/MaconomyPortal.orctrain.US_MCS.exe/

   Examples of links pointed to client-specific translation files:

   Java: http://serp02.orckestra.local:81/cgi-bin/Maconomy/Jaconomy.orcktest.en_US_ORC.exe/Maconomy/JaconomyMP/webstart.jnlp

   Portal: http://serp02.orckestra.local:81/cgi-bin/Maconomy/MaconomyPortal.orcktest.en_US_ORC.exe/

   If your links do not point to client-specific files, select the desired languages in the Web Products screen in MConfig.

Update the Translation Files

Translation files are kept in two locations:

- **Application server** (java)
- **Web server** (Portal)

Whenever you change the translation file, you must update it in BOTH places. Additionally, each environment must be updated separately.

Server Locations

App Server

D:\maconomy\<*version*><*env*>\MaconomyDir\Definitions

where

<*version*> represents the version of Maconomy installed at the client, and

<*env*> represents your current environment

Example: D:\maconomy\w_12_0.test\MaconomyDir\Definitions

Web Server

D:\maconomyweb\<*shortname*>\cgi-bin\Maconomy

where

<*shortname*> represents the shortname

Example: D:\maconomyweb\orcktest\cgi-bin\Maconomy

Before you begin, it is important to note that each and every new instance of a word must be translated as a separate string. For example, to translate Job to Project, you must update the word in each of its forms and each time it occurs in a different string.

You can either **add a new term** (if it is not in the translation to begin with) or **update an existing term** (if it is in the file but you want it to be translated to something else). See details below.

To update the translation files, complete the following steps:

1. Copy and paste the custom translation file from one the App server (such as : e.g. c:\Maconomy\w_17_0.orctest\MaconomyDir\Definitions\) onto your computer.
2. Open the document either a text editor (Notepad++ or Crimson Editor), keeping the original file format.

   We strongly discourage the use of Excel as it often automatically and incorrectly changes whitespace and special characters.

3. Delete the stamp, which is all leading lines starting with two hash marks(usually the first six lines), such as:

   ##505783
   ##encoding = UTF-8
   ##VERSION W 17.0 p102
   ##USERLANGUAGE en_US_ORC
   ##CHARACTERSET Unicode
   ##MOTV Mon Oct 26 12:07:29 2015

4. On the left hand side are the “source” (From) strings to be translated, on the right are the “target” (To) strings. Add or update terms as needed.

For example:

Source	Target Column
Job	Project
The job <x> is in use	The project <x> is in use
job	project
jobs	projects
create new job	create new project

5. Stamp the translation file.
6. Move the stamped translation file back to the server (and web server).
7. Place the stamped file on both the Maconomy Server and the Web Server.

Add a New Term

To add a new term, complete the following steps:

1. Search for the term you want to add, to confirm that the term does not already exist in the file.
2. Add the new term to the bottom of the translation file.

Since it is difficult to find every occurrence of new term throughout the system, usually this is an iterative process, as you find all lines where your terms appear and add to your dictionary.

For more information, see “Getting Started with Localization” where the process is explained and tooling (the Solution Dictionary program) is described.

Update an Existing Term

1. Use **CTRL+F** to find the term you want to change.
2. Update the target column. Replace the existing translation with the new translation (using care not to change the From/source column.

   When you update terms, you are more likely to find every occurrence of the word/term because a developer has already identified them, or the term has been refined so many times that all the occurrences have been discovered.

If this is the first time you’ve ever worked on a translation file, start by making couple of changes, then validate and use the online validation tool to make sure the file works properly. Then, add more changes as needed.

Stamp the File

Use an online validation tool to stamp the translation file:

http://dictionary.maconomy.com/cgi-bin/motv.pl

Username: Partner

Password: DitfN0p

This is an internal tool available to our consultants and partners – NOT for customers directly.

Once you're logged in, follow the step-by-step instructions, selecting from the drop-down options as detailed on the site and below. The site also features a number of documents with more information.

To stamp / validate a file, complete the following steps:

1. Select the appropriate version from the **Choose Version** drop-down.
2. Select user language (For version 15.0 and below: FR=French, ES=Spain, and so on. For version 16.0 and above, fr_FR=French, es_ES=Spain, and so on.)
3. Select the character set (**Standard** for version 15.0 and below, and **Unicode** for version 16.0 and above).
4. Select encoding, such as ISO-8859-1 or UTF-8. This should match the characterset used when saving the dictionary.
5. If your translation file is based on one of the Solution translation files (for example, en_US_CPA or en_US_MCS), select Solution. Otherwise, select **No Solution**.
6. For custom translation files, enter the customer designation, such as ORC (uppercase). It becomes part of the dialect and part of the language name. Leave the Custom solution blank if you are creating a translation file for a language not delivered with Maconomy.
7. Select **Complete dictionary**, as follows:
    - Select the check box if the translation file contains all the terms in the system. Usually this is checked only when creating a dictionary for a new language.
    - Do not select the check box if if the dictionary is based on a standard or solution dictionary (i.e. only contains custom translations).
8. Browse to choose your translation zip file.
9. Select **Convert to Unix format** check box if the customer's Maconomy Server is a Unix server. Otherwise, do not select the check box.
10. Select **Do extra Quality check** check box for extra validation of the translation file. This produces a report with warnings and notifications regarding possible issues with the translations, such as duplicate translations, translations with wrong placeholders, and so on. Deltek recommends selecting this option.

    US / UK users must modify the time stamp. See details below.

11. Click **Upload**. When complete, download the zip file back onto your computer.

Modify the Time Stamp

Since the validation tool is in Denmark, for US / UK users, the date and time of the stamp might be in the future (six hours for US, 1 hour for UK). If this happens, you must reset the date modified property before moving the file back onto the servers, otherwise you will receive an error when accessing the link.

Right-click a file and select Properties to see Created and Modified properties, as well as time stamps on the file.

To modify the time stamp, complete the following steps:

1. Shift-Click the folder where the stamped translation file is located and select “Open command window here”
2. On the command line, enter the following (replacing the translation file name):

   ```
   Copy /b Translations.W_17_0.en_US_CPA+,,
   ```

   Note the plus [+] and two commas [, ,] at the end.

Move File Back to Server

To move the file back onto the servers, complete the following steps:

1. Confirm that no one is logged into the environment you are going to update.
2. Shut down the webdeamons and coupling services on the App server (under **Start > Services**).
3. Re-confirm that no users are logged on (web daemons should disappear).
4. Copy the stamped translation file to **MaconomyDir\Definitions** on the App server.
5. Copy the stamped translation file to **cgi-bin\Maconomy** on the web server.
6. Delete the “.hash” files related to the translation file you are updating in MaconomyDir\Definitions on the App server.
7. Delete “.hash” files in cgi-bin\Maconomy on the web server.

Confirm Users are Off Server

To ensure users are no longer logged into the environment, complete the following steps:

1. Open the command prompt.
2. Run the Maconomy server executable with the ‘--users’ option against the chosen Maconomy system:

   MaconomyServer.exe -i<system-name> -S<shortname> --users

   or:

   macoracle.<system-name> -S<shortname> --users

   where

   <*system-name*> is the name of the Maconomy system.
   <*shortname*> is the database shortname to connect to.

   Example:
   MaconomyServer.exe -imaconomy -Smacoprod --users
3. Click **Enter**.

   A list of logged on users displays. If users are still logged on, see steps in the next section.

Tips & Tricks

If Users Are Still Logged On

If users are still logged on when you need to move files back to the server, follow the steps below to bump people off the server. Stop the service altogether to prevent people from getting back on the server, as needed.

To bump people off the server, complete the following steps:

1. Double-check that no Coupling Services are running.
2. Run the Maconomy server executable again with the following options to remove all user sessions: **--users:remove=all**

If there are still users logged on, seek advice from a (more) senior consultant.

.Hash Files

.hash files are temporary files that are created the first time a translation file is accessed. You must delete the old one so that a new one can be generated by the new translation file you are about to install.

See the Move File Back to Server section for further steps and locations.

.Lay Files

In order to translate a word in the Portal, it must be marked with **#T** in the**.lay** file.

If you see a word in the Portal that seems like it should be translated but is not, the problem might be in the .lay file, not the translation. Add **#T** (see below) to the word in the .lay order for the system to recognize it.

In the .lay files for the portal, you will notice the following “tags” on each field, which tell the system whether to translate the string in the portal, as follows:

- #N is used to indicate no localization. This is used for portal framework keywords like "component" or "input."
- #T is used to indicate template localization, meaning that the text is translated to user's language, such as. the language in the URL of the Portal link.
- #K is used to indicate kernel localization used for internal Maconomy identifiers, such as table, field, or dialog names.

Quotations in Text

How texts are handled depends on where they appear. There can be issues if you use quotations in the middle of a phrase. Layout terms seem to be sensitive to quotes, but error messages are not. If you need quotes in layouts, use single quotes.

Text in Curly Braces

Text in curly braces (for example, {No Translation}) appears in the translation file, but is not displayed for users. This text is used for two purposes:

- As a hint to you, the translator, from Engineering, to give a hint as to the intended meaning of a term.
- To indicate two uses of the same unique term, such as “Invoice,” which can mean both “to invoice” or “an invoice,” and which has different translations in Danish.

Maconomy does not display anything inside curly braces, so your translation (after the tab character) can have anything you like inside, or you can leave them out altogether.

In this case, {No Translation} is probably a hint that the term should not be translated, such as a term for demo data.

In other cases, it might be a technical term which is used by code which will stop working if it is translated.

Workspace Client Gender Terms

Workspace Client terms are added to the MOTV Dictionary validation site to accommodate gender specifications.

To indicate the list of possible genders, translate the term:

|Gender|{gender}

To indicate that the target language has (M)asculine, (F)eminine, and (N)eutral genders, enter:

|Gender|{gender} |M|F|N|

To indicate that the target language has only one gender, enter:

|Gender|{gender} |N|

To indicate the gender of a term, specify the correct gender on the line. For example, to indicate that an A/P Entry is of Masculine gender, add the following line:

Gender{A/P Entry} M

Portal / Java Clients – Existing Terms

The Workspace Client method is similar to the existing way of specifying gender in Portal / Java Clients:

Kreditoreneintrag@Gender M

Note that the existing term uses the translated term (Kreditoreneintrag) but the new term is English (A/P Entry).

Encoding

Character encoding is used when saving the dictionary file in the text editor as opposed to the character set used on the server. The default so far has been Western European ISO-8859-1, Latin Alphabet No. 1, but the Workspace Client supports the use of other character sets.

Currently, only single-byte character sets are supported.

Encoding can be specified by selecting an encoding in step 4 of the validation tool or directly in the dictionary by a line with the encoding specification:

##encoding = ISO-8859-1

Automatic Translations

Many of the new terms for the Workspace Client can be translated automatically with a few prerequisites:

Translate the gender term: |Gender|{gender}

Translate a few generic, gender dependent terms:

|Add ^1|

|Create ^1|

|Delete ^1|

|Insert ^1|

|Move ^1 down|

|Move ^1 up|

|New ^1|

|Print ^1|

|Save ^1|

For example:

|New ^1| |Neuer ^1|Neue ^1|Neues ^1|

To get the suggested new translations, validate the dictionary by selecting the **Do extra Quality check** check box. When downloading the result, the zip file will include a new _**Suggested.txt** file with possible translations for missing terms. Verify the suggestions, add them to the dictionary, and make a new validation.

To efficiently automatic as many translations as possible, complete the following steps:

1. Translate the terms mentioned above.
2. Validate the dictionary by selecting the **Do extra Quality check** check box. This should suggest translations for many gender terms like 'Gender{Entity}'. These suggestions are based on the existing @Gender entries.
3. Verify the suggestions and add them to the dictionary.
4. Validate the dictionary to get a list of missing gender terms.
5. Translate any remaining gender entries
6. Validate the dictionary. This should suggest translations to the generic terms above.

For example:

New A/P Entry	Neuer Kreditoreneintrag
New A/P Reconciliation	Neue Kreditorenabstimmung

Tools and Resources

Depending on whether you are creating a new dictionary from scratch or you are updating an existing dictionary, and depending on your experience with the Maconomy system, the translation can be a large and complicated project. Strings in the dictionary are out of context and may contain special comments, abbreviations, and notations that need to be explained. Deltek therefore provides you with a number of tools and resources to help you in your translation work.

Access the tools and resources from the Localization Centre:

https://home.deltek.com/sites/Maconomy/LocalisationCentre/default.aspx.

Term-in-Aider – Context Finder

The correct translation of a word often depends on the context in which it is used. This information is not available in a dictionary. In order to ensure the correct terminology, you can look up the strings using the Term-in-Aider which brings up a list of the windows and scripts in which the current string is used. Layout definition files can also be displayed for even more context details (such as the island in which a given field appears). Furthermore, for strings used in window layouts for the Java client, you can go directly to the online help for the needed window for an explanation of the use of that window and field. Lookups are made quickly and easily by using a few keystrokes.

Solution and Custom Dictionary Builder

When creating a custom dictionary, finding the lines to be modified, moving them to the custom dictionary and modifying the lines can be a tedious task, especially when starting for the first time.

The Solution Dictionary tool is created to help with this process. This program finds lines in a dictionary that match a list of terms and in those lines, the standard translation of the original term is replaced with the solution term.

The input to the program is:

- A file with terms to change
- A standard (or solution) dictionary with translations to change
- A solution (or custom) dictionary with changed translations

The program creates a file with suggestions for new lines to the custom dictionary.

Following is an example used in the Danish solution dictionary:

```
Job    Job    Projekt
```

This line will handle lines containing “Job” and replace the standard translation of “Job” with “Projekt”.

In some cases, however, we want to exclude some “false positives.” The next example is more complex:

```
(?<!Id)Entity Entity Udførende afd.
```

This line uses regular expressions to handle lines containing "Entity" but not "Identity". Again, the standard translation of "Entity" on those lines will be replaced by "Udførende afd."

For tips on using regular expressions see:

http://www.regular-expressions.info/quickstart.html

The replacement is intelligent about casing and contracting words, but it is not perfect, so the output file should be carefully reviewed before adding it to the custom dictionary.

How to…

Use the Custom Program

To use the custom program, complete the following steps:

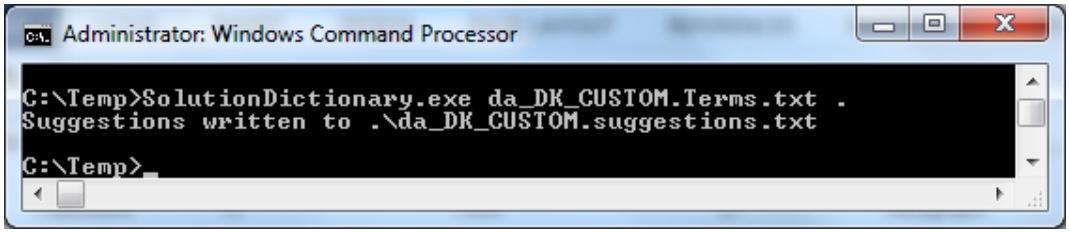
1. Copy the standard dictionary to a working folder and rename it, such as:da_DK_total.txt

2. If an existing custom dictionary is being updated, copy it to the same folder and rename it, such as. da_DK_CUSTOM_total.txt (ensure that it contains the same prefix and postfix as the standard dictionary).
3. Create a term list file with instructions on what to change, such as da_DK_CUSTOM.Terms.txt, in the working folder.

The format of the list is three tab separated columns:

- Search expression, wherelines in the dictionary matching this regular expression are handled (see above)
- Original English term
- Custom translated term

4. Consider variations of terms (opportunity/opportunities)
5. Create a file with suggestions for the custom dictionary.
6. Run one of the following command to create <OutputPath>\Suggestions.txt:
   `SolutionDictionary da_DK_CUSTOM.Terms.txt <OutputPath>`
   `SolutionDictionary -c s da_DK_CUSTOM.Terms.txt <OutputPath>`
   The second form will attempt to take the “connection character” ‘s’ in e.g. sag***S***fremdrift into consideration.



7. Review the contents of Suggestions.txt before adding them to the custom dictionary.

Add New Terms

To add new terms to the Terms.txt file, complete the following steps:

1. Update the translations in the existing custom dictionary by creating a Terms.txt file containing only the new terms and run the program on the existing custom dictionary:

   `SolutionDictionary da_DK_CUSTOM.Terms.txt <OutputPath>`

2. Add the terms to the existing terms file and replace any relevant suggestions in the existing dictionary.

After Translation

When you have translated a list of new strings for a dictionary, add the translated lines to the latest version of the existing dictionary (if any) and stamp it using the MOTV tool. Part of the stamping procedure includes a validation to ensure that the system will run using the updated dictionary (that e.g. no two lines have the same literal translation or use illegal characters). To upload a dictionary for stamping, place it in a zip file containing no other files and follow the instructions of the MOTV tool.

If any problems occur in the validation, they are listed in the log files returned by MOTV. You will receive a list of the lines causing the validation to fail, and you can make the necessary adjustments. Errors must be fixed before the dictionary can be used. Warnings should be reviewed to ensure that the reported issues are acceptable. See the document “Fixing typical errors and warnings in translation files” for information about resolving these errors.

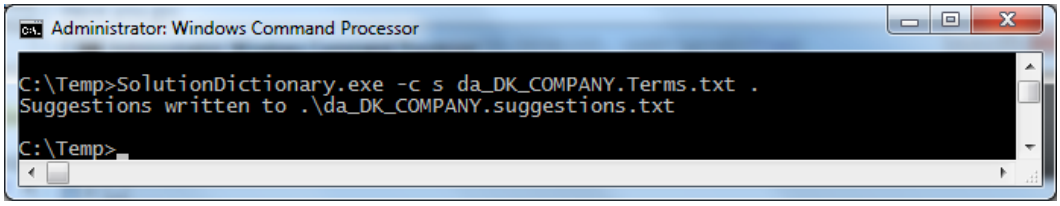
To use the program, complete the following steps:

1. Copy the standard dictionary to a working folder and rename it, such as da_DK_total.txt.
2. If an existing custom dictionary is being updated, copy it to the same folder and rename it to da_DK_COMPANY_total.txt, for example (same prefix and postfix as the standard dictionary).
3. Create a term list file with instructions on what to change, such as da_DK_COMP.Terms.txt in the working folder. The format of the list is three tab-separated columns:
   a. Search expression - lines in the dictionary that match this regular expression will be handled (see above).
   b. Original English term.
   c. Custom-translated term.
4. Consider variations of terms (opportunity/opportunities).

   Create <OutputPath>\Suggestions.txt: run one of the following commands to create a file with suggestions for the custom dictionary:

   ```
   SolutionDictionary da_DK_COMP.Terms.txt <OutputPath>
   SolutionDictionary -c s da_DK_COMP.Terms.txt <OutputPath>
   ```

   The second form will attempt to take the “connection character” ‘s’ in, for example, sagSfremdrift, into consideration.



Documentation

The first thing that you need to do before starting work on a missing list is to read the documentation on the procedures, guidelines, and requirements for formats and notation. Currently, the following documentation is available:

- **Dictionary Format and Notation manual** — The strings in the missing list originate from different sources in the Maconomy software, not just windows and printouts. Error messages, menu titles, and tool tips are also part of the dictionary, in addition to import programs and reports. You will therefore encounter placeholders, special characters, and notations that need to be handled in a certain way.

  This section describes the most common types of notations and line types, thus making it easier for you to achieve the correct notation in the translation. This section also describes how genders are specified and how to manage gender-dependent grammar in the dictionary.

- **Maconomy Reference Manual/Online Help** — The reference manual/online help provides detailed information about the functionality of the Maconomy system and the Portal components.
- **Product Information** —The Product Information describes the new features that are introduced from one version of Maconomy to the next. By reading the product information for the version whose missing list you are translating, you become acquainted with the functionality behind the strings in the missing list, which is a shortcut to deciding on the correct terminology.

Clients

Java Client

Customizing the Special Menu

This document describes the configuration of the Special… menu item in the Analyze menu of the Maconomy client.

Customizing the Special… Menu Item

The Analyze menu in the standard Maconomy client contains an item called Special.... By selecting this item, you can select and run a custom Analyzer or SQL report, and you can run reports based on Maconomy Universes.

If you modify a file and place the Analyzer and SQL reports on the Maconomy server, all users of Maconomy can access these reports through a simple menu (if they are granted access through the standard access control windows).

How to Customize the Special… Menu Item

To make a report available to users, first copy your compiled custom Analyzer or SQL report to the `Analyze` folder on the Maconomy server. Then create or edit the Special.txt file, also in the `Analyze` folder.

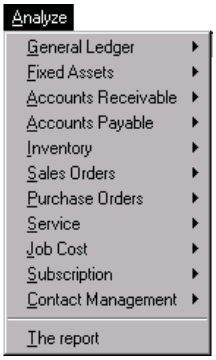
For each custom report, add a line to the Special.txt file:

```
GRAPH report.grf "The report"
```

This example defines that the Analyzer report report.grf should appear on the Analyze menu as a menu item labeled “The report”. Item one in the line, GRAPH, is a prefix that identifies custom report menu items. Item two is the file name of the custom report, and the last item is the text that appears in the Analyze menu.

To test the new Analyze menu, open the Maconomy client. The Maconomy client then updates the access control information to match the contents of the Special.txt file. When the Maconomy client has finished updating the access control information, restart the Maconomy client, and log in to Maconomy.

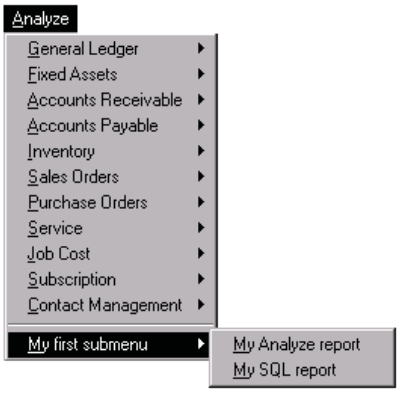
The Analyze menu and the Groups window in the Set-Up module now contain the custom reports that are specified in the Special.txt file. The custom Analyzer report now appears at the bottom of the Analyze menu as shown in the following figure.



If the number of custom reports becomes large, you can structure the list of custom reports using submenus, just like the standard Analyze menu. For each submenu, enclose the GRAPH lines like this:

```
SUBMENU "My first submenu"
   GRAPH report1.grf "My Analyze report" GRAPH report2.grn "My SQL report"
END
```

This example defines a submenu on the Analyze menu, labeled “My first submenu”. The Analyze menu now appears as shown in the following figure.



Other Options

Add a Separator Line

If you want to add a separator line between reports in the menu, add the keyword

```
LINE
```

on a separate line in the special.txt file.

Store Analyzer Reports Locally

You can enable the users of the Maconomy client 3.*x* for Windows and for Macintosh to open a dialog and search for a locally stored Analyzer report by adding the line

```
GRAPH $Special "Special..."
```

in the special.txt file. This adds a “Special...” menu item to the submenu, which opens a file selection dialog. Using this dialog, users can select and run a custom Analyzer report.

Add Universes

You can add a universe to the Special … menu of the Maconomy client 4.*x* for Windows. In the Analyzer, you can select fields from the added custom universe. Note that to add a universe, the universe must be installed using MBuilder on the Maconomy server (see the MUL Reference and MBuilder Reference manuals for more information).

For each universe to be made available in the Special… menu, add a line to the special.txt file similar to the following:

```
UNIVERSE JobCost::Universes::TaskU "Standard Task Universe"
```

This example defines that the universe JobCost::Universes::TaskU appears in the Analyze menu as a menu item labeled “Standard Task Universe.”

Note that the menu item label, here “Standard Task Universe,” must be unique for all items in the Analyzer menu.

Shortcuts in the Java Client

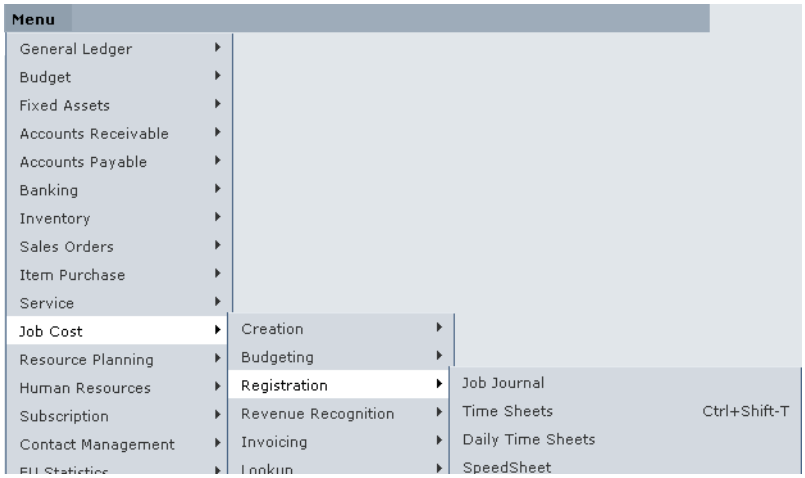
With Maconomy version X, the ability to add shortcut keys to windows, reports, and actions in the Java client was added.

You define the shortcut keys in an XML file on the server. When the client is launched, the shortcuts are read from the file and applied to the client.

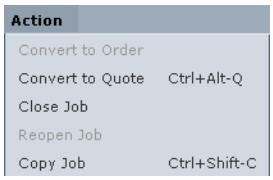
If you “double book” a shortcut key, that is, if you define a key as a shortcut that is already in use as a shortcut by Maconomy or the operating system, the key that you defined will not be applied as a shortcut.

Visibility

When a shortcut key is defined correctly as specified in the following section, it appears in the appropriate menu of the Java client. In the following figure, the shortcut CTRL+SHIFT+T has been defined to open the window Time Sheets.



In the following figure, shortcuts have been defined for the Convert to Quote and Copy Job actions. The following Action menu is from the Jobs window:



Note that you must restart the Java client to see any changes that you make to the shortcut definition file.

If the same shortcut key is defined more than once in the file, the last one to be defined prevails.

Defining Shortcut Keys

Shortcut keys are defined in the shortcuts.xml file, which is placed in the following location: <MaconomyDir>/Jaconomy/. This file contains examples of shortcuts that have been commented out (so that they are not in effect), but you can use the file for reference. The XML schema

definition file shortcuts.xsd is in the same directory; you can validate the shortcuts.xml file against the shortcuts.xsd file in an XML editor.

Available Keys

The following keys can be used to define shortcut keys: Alphabetic letters a-z and A-Z and numerals 0-9.

Note that function keys (F1-F12) cannot be used as shortcut keys.

The following keys can be used as modifier keys, that is one or more keys that must be pressed together with the shortcut key: Shift, Ctrl, and Alt (or a combination of these). Note that on the Macintosh, the Ctrl key is mapped to the Apple Command (💻🖉💻) key.

Structure of the shortcuts.xml File

The following describes the elements of the shortcuts.xml file.

Header

The file header should be as follows:

```
<?xml version="1.0"?>
<types:shortcuts
   xsi:schemaLocation=http://maconomy.com/types shortcuts.xsd
   xmlns:types=http://maconomy.com/types
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
   <!—
      Key definitions go here
   -->
</types:shortcuts>
```

Shortcut to Open a Dialog Window

```
<shortcut
   Key="T" SHIFT="true" CTRL="true"
               ALT="false"
   Dialog="TimeSheets"/>
```

In this example, the shortcut key CTRL+SHIFT+T opens the Time Sheets window. The dialog is defined by its internal name. (You can find the internal name of a window by looking at its MDL layout exported from the Window Layouts window.)

Shortcut to Open a Report

```
<shortcut
   Key="R" SHIFT="true" CTRL="false"
   Alt="true" Reort="A
   ::UA::INTERCOMPANYSVIE"/>
<shortcut
   Key="A" SHIFT="false" CTRL="true"
               ALT="true"
   Report="Activities.msl"/>
```

In the first example, the shortcut key Alt+Shift+R opens the MQL report with the internal name A ::UA::INTERCOMPANYSVIE.

In the second example, the shortcut key CTRL+ALT+A opens the legacy report Activities.msl. The report must be specified in the file Special.txt in the Analyze folder. For more information, see “Customizing the Special Menu”.

Shortcut to Execute an Action

```
<shortcut
   Key="C" SHIFT="true" CTRL="true"
   Alt="false" Action="CopyJob"/>
```

In this example, the shortcut key CTRL+SHIFT+C executes the action with the internal name CopyJob in all windows where this action is available. Note that action shortcuts can also be specified in the MDL layout for any dialog.

Unavailable Keys

If there is a clash between a shortcut key defined here and a shortcut key defined by Maconomy or the operating system, the shortcut key defined here is not available.

Currently, Maconomy does not verify the contents of the shortcuts.xml file when loading it. You should therefore check the following list of shortcut keys to see whether the shortcut key that you intend to use is available.

Key	Windows	Macintosh	Description
0	Alt+0	Option+Shift+0	Bring menu window in front
0	Ctrl+0	💻🖉💻0	Actual size
1	Ctrl+1	💻🖉💻1	First record or page
2	Ctrl+2	💻🖉💻2	Previous record or page
3	Ctrl+3	💻🖉💻3	Next record or page
4	Ctrl+4	💻🖉💻4	Last record or page
5	Ctrl+5	💻🖉💻5	First table line
6	Ctrl+6	💻🖉💻6	Previous table line
7	Ctrl+7	💻🖉💻7	Next table line
8	Ctrl+8	💻🖉💻8	Last table line
A	Ctrl+A	💻🖉💻A	Select all
C	Ctrl+C	💻🖉💻C	Copy
C	Ctrl+Shift+C	Shift+💻🖉💻C	Open column selection tab
D	Ctrl+D	💻🖉💻D	Delete record
F	Ctrl+F	💻🖉💻F	Open find pane
G	Ctrl+G	💻🖉💻G	Foreign key search
H		💻🖉💻H	Hide Maconomy
H		Option+💻🖉💻H	Hide Others
H	Ctrl+Shift+H	Shift+💻🖉💻H	Search criteria builder
J	Ctrl+J	💻🖉💻J	Open related window
L	Ctrl+L	💻🖉💻L	Delete line in table
M	Ctrl+Shift+M	Shift+💻🖉💻M	Insert line in table
M	Ctrl+M	💻🖉💻M	Add line to table
N	Ctrl+N	💻🖉💻N	New record
O	Ctrl+O	💻🖉💻O	Open report in Report Component
P	Ctrl+Shift+P	Shift+💻🖉💻P	Print this
P	Ctrl+P	💻🖉💻P	Print or Paste (in text fields)
Q	Ctrl+Q	💻🖉💻Q	Exit Java Client
R	F5	💻🖉💻R	Refresh, get data from server

R	Shift+F5	Shift+💻🖉💻R	Revert, clear unsaved changes
R	Ctrl+R	💻🖉💻R	Run report in Reporting Component
R	Ctrl+Shift+R	Shift+💻🖉💻R	Open row selection tab
S	Ctrl+S	💻🖉💻S	Save data or report to server
V	Ctrl+V	💻🖉💻V	Paste in text fields
W	Ctrl+F4	💻🖉💻W	Close window
X	Ctrl+X	💻🖉💻X	Cut in text fields
Y	Ctrl+Y	💻🖉💻Y	Redo in text fields
Z	Ctrl+Z	💻🖉💻Z	Undo in text fields

Maconomy Extender

The Maconomy Extender is an integrated development environment (IDE) that is dedicated to customizing Maconomy. The Extender allows business consultants or technical consultants to change components or develop new components, to keep these under version control, and to deploy these components to test and production systems.

For detailed information about the Maconomy Extender, see the Maconomy Extender manual.

RGL Report Designer Overview

This document provides information about the RGL Report Designer.

The RGL Report Designer

The RGL Report Designer is a Windows program used to compile RGL Reports source (.gr files) into executable RGL Reports (.grn files).

Getting access to the RGL Report Designer program for Maconomy 2.0.2 (and later versions) requires access to the Maconomy Application Server.

Using the Maconomy Application Server, you can create a .zip archive that packages the complete kit for compiling RGL Reports using the RGL Report Designer program.

The RGL Report Designer .zip file is generated using the following Maconomy Server option:

```
--CreateRGLReportDesigner <pathToZipFile>
```

Examples of using this command include:

On Linux and UNIX:

```
/data/maconomy/bin/MaconomyServer.<systemName> -S<shortName> --
CreateRGLReportDesigner /tmp/MyReportDesigner.zip
```

On Windows:

```
C:\maconomy\bin\MaconomyServer.<systemName> -S<shortName> --
CreateRGLReportDesigner C:\Tmp\MyReportDesigner.zip
```

Remember to use a "work" folder as the destination for the .zip file (for example, "/tmp").

After creating the .zip file, transfer it to the Windows machine(s) that needs to use the RGL Report Designer program. Unpack the .zip file and run the included **ReportDesigner.exe** file. There are no install or uninstall programs required; the RGL Report Designer kit is self-contained. To remove the RGL Report Designer from a Windows machine, move the RGL Report Designer kit to the trash.

In addition to the RGL Report Designer program, the RGL Report Designer kit contains documentation related to reporting in Maconomy 2.0. Look in the Documentation folder next to the **ReportDesigner.exe** program.

Note that this RGL Report Designer generated by the `--CreateRGLReportDesigner` Maconomy Server option is derived from the TPU and APU versions used by the `<systemName>` above.

- If the TPU is updated, it might (although unlikely) introduce changes that would require a new RGL Report Designer to be generated.
- If the APU is updated, it might introduce additional table columns and/or tables (more likely). This would require a new RGL Report Designer to be generated and the RGL Reports to be recompiled.

Check the Release and Installation notes after installing a TPU and/or APU for information related to the RGL Report Designer.

Access Control View Templates

Only two templates are currently available, but there are plans to expand the template framework with the addition of more templates and Oracle support. Due to database-specific optimizations, templates must be different for each database. Only the Microsoft SQL Server database is supported.

The union template is a rewrite of original Maconomy access control views. It replaces subselects with union operators since it is less costly to perform a union on the SQL server. In Oracle, this is not the case, thus the need for database-specific templates. The union template is a dynamically generated template and can be applied to all views matching a specific signature in the RelationAccessDefinitions file (located in the Maconomy application directory). Depending on how access control is set up on any given Maconomy system, MConfig generates a list of views that can use the union template.

The noemptykey template, on the other hand, can be applied only to specific tables where business logic ensures that key values for a table are never empty. Therefore, the list of allowed views is static.

When to Use…

The best-performing setup is a combination of the following:

- Applying the noemptykey template to the acjobentry view
- Applying the union template to all other views (that can utilize this template)

This may vary depending on how the data is distributed in a given database. Be aware that applying the union view can have a negative effect on performance for the Administrator user (or for any user with equal or similar access rights), as illustrated by the test results for the ACJobentry table in the Findings section. If this issue occurs, use the noemptykey template on the table (if applicable), or revert to the stock view for that table.

Findings

This section illustrates performance differences between the views for two tables that, in most Maconomy installations, contain the largest number of records. Test results are shown for the following:

- ACFINANCEENTRY — A table containing 16 million entries

- ACJOBENTRY — A table containing five million entries

The three test types used are the following:

- Wsc (Multiple) — This involves opening the corresponding job or finance entry list in a workspace and measuring the time it takes to populate the first page.
- Direct Fresh — This is a cold (that is, not cached) run against the database with the same query that is used in the Workspace Client to populate the table.
- Direct Multiple Runs — This involves executing the same query as in Direct Fresh, but multiple times. The goal is to check whether caching has an impact on performance.

Time is measured in seconds. “0s” means that the query is executed in less than a second. Actual execution time in milliseconds is no longer displayed because, to the Client user, the time difference is no longer distinguishable.

The rows in bold font indicate the best template to use for each view in the test database.

ACFINANCEENTRY

Template	Wsc (Multiple)	Direct Fresh	Direct Multiple Runs
User with limited access			
default	46s	106s	45s
noemptykey	45s	106s	45s
union	**9s**	**9s**	**9s**
Administrator user			
default	1s	1s	1s
noemptykey	1s	1s	1s
union	**1s**	**1s**	**1s**

ACJOBENTRY

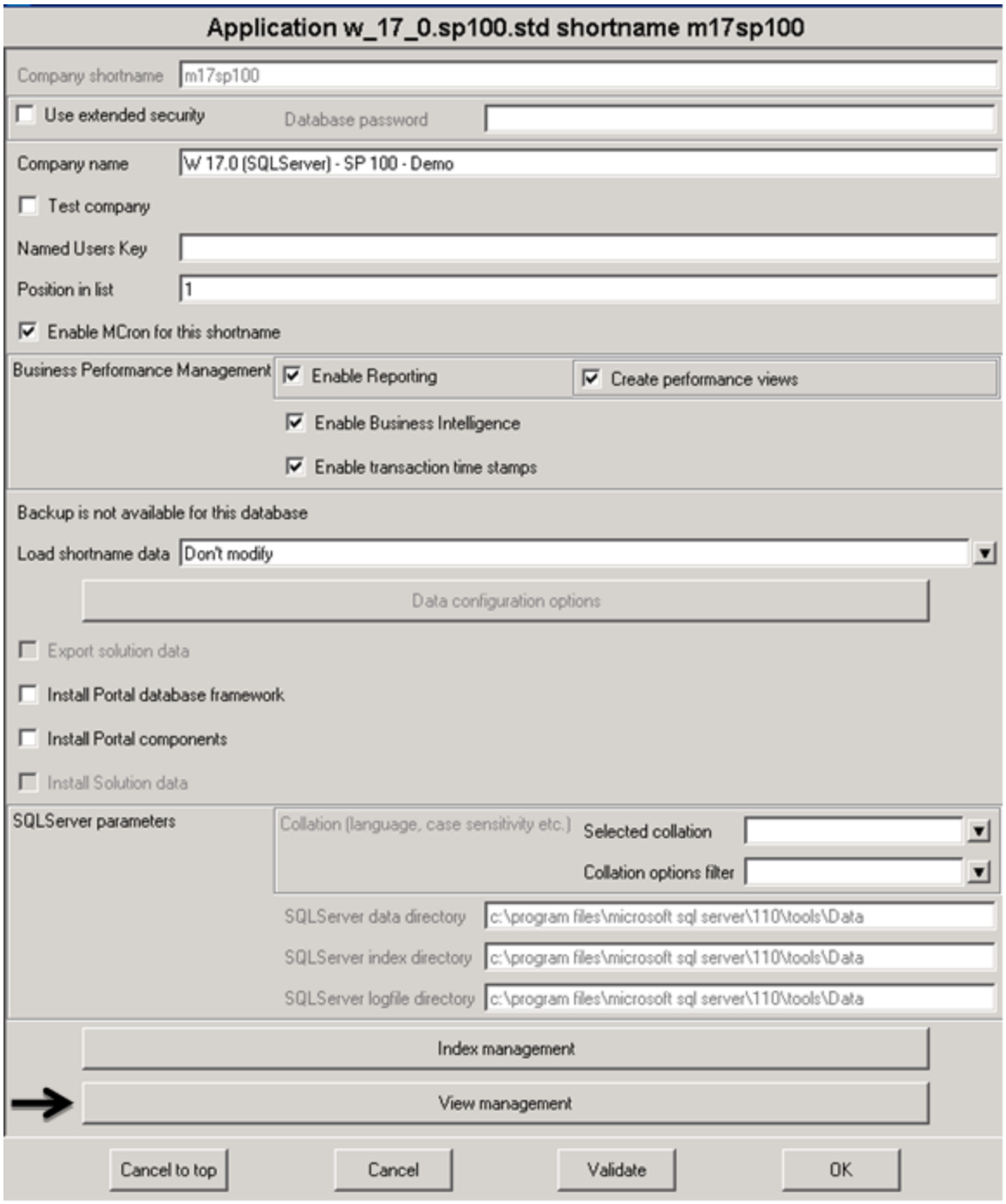
Template	Wsc (Multiple)	Direct Fresh	Direct Multiple Runs
User with limited access			
default	36s	40s	35s
noemptykey	**1s**	**0s**	**0s**
union	1s	1s	1s
Administrator user			

Template	Wsc (Multiple)	Direct Fresh	Direct Multiple Runs
default	1s	0s	0s
noemptykey	**1s**	**0s**	**0s**
union	180+s	180+s	180+s

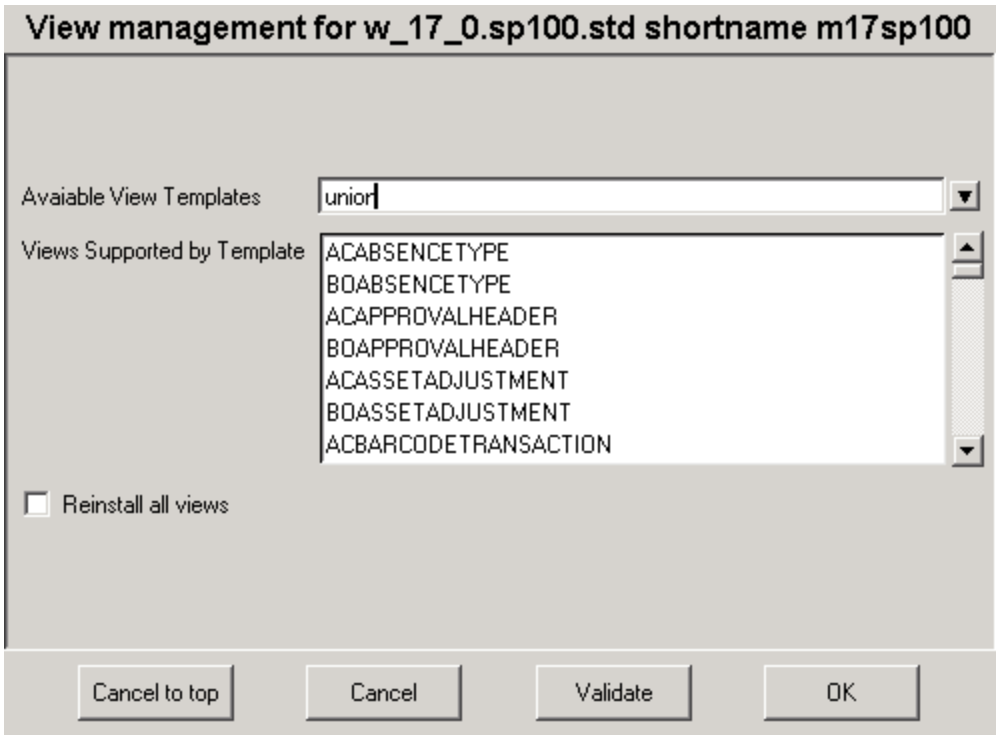
Setup

As of MConfig version 8.6, Access Control view templates can be controlled from a View management window that can be accessed through the application shortname window in MConfig.

The following figure shows the application shortname window.



The following figure shows the View management window.



The View management window allows you to select which template to apply to specific views. Note that the templates do not support all views. If you select a template, the window displays the views that are supported for that template. Since all view changes are completely reversible, and the updated views function in the same way as the original views, faster performance should be your only consideration when applying a template to a supported view.

If you make any changes to access control in the Maconomy system (specifically, to the RelationAccessDefinitions.txt file), you need to reapply the AC view templates through MConfig to ensure the validity of the selected configuration.

Access the View Management Window

To access the View management window, complete the following steps:

1. Start MConfig.
2. Double-click an application from the list box to open the application window.
3. Double-click the shortname to open the shortname window.
4. In the shortname window, click **View management**.

Reapply the AC View Templates

To reapply the AC view templates, complete the following steps:

1. In the View management window, select the **Reinstall all views** check box.
2. Click **OK**.
3. If the views are reinstalled with no errors, apply the desired view template configuration.

MDoc

MDoc is a tool for generating a set of hyperlinked HTML files that offers a database description for each field and action within Maconomy. Information for each field/action includes: (database) name, type, (user interface) title of field, and other information.

How to Generate MDoc

To generate MDoc, complete the following steps:

1. In the bin folder of the TPU, click the MDumper program to generate the files on any Maconomy installation from the application server.
2. Create an output folder in any location where you have write access.

Be sure to create the Output folder prior to running the command below.

3. Use the following command to generate files using MDoc:

   `MDumper -MaconomyDir <application home> -DumpMDoc <output folder>`

   where

   <application home> is the application home for the targeted application

   and

   <output folder> is the folder where the generated MDoc files are placed.

Instructions are maintained in the System Administrator Guide documentation suite.

How to Use MDoc

After you generate MDoc, use the tool to look up database items and view descriptions.

To use MDoc, complete the following steps:

1. Run the tool to generate a zip file.
2. In the output folder you created, extract the contents of the zip file.
3. Double-click the MDoc folder.
4. Double-click the html folder.
5. Double-click **index.html** to open the MDoc start page in the internet browser.
6. Select a database item from the menu to view its description.

Standard Import Programs Overview

This document describes the use of import programs for Maconomy 2.0+. Before you proceed, make sure that your import program and program specific import manual match the same Maconomy version and service pack number as this document, as a mismatch may cause errors in your import. This document describes the central concepts involved in the import process and explains the possibilities offered by import programs. In addition, the manual sets up a number of examples for the practical use of import programs. The concept definitions and examples are described on a general level.

Please note that the functionality described in this manual only applies to those Maconomy import programs that are generated according to the Standard Import Format. Maconomy's product range also includes a number of import programs with functionality that is not covered in this section.

Introduction

How to Read This Section

This section consists of two parts. The first part describes the standard functionality of Maconomy import programs, including technical details, data allowed in an import file, and a list of concepts related to data import. The first part thus acts as a technical handbook describing the exact functionality and explaining how import programs interpret different information.

The second part is a guide to creating your own import files for different purposes, and to using the tools available in the import programs in practice. The second part describes the work flow of setting up an import file and contains step by step lists which you can follow when setting up your import file.

After you have read this section, you can apply the general functionality described to any Maconomy standard import program. Refer to "Getting Started" to begin.

Getting Started

This section briefly describes how to get started working with import programs. More detailed explanations are given in subsequent chapters.

Specific import file manuals consist of lists of line types available in the program. Each line type includes a list of available fields.

To work with a specific import program, you must:

- Activate Help. This action produces a file that lists the available line types.
- Activate printformat. This action produces the fields for the corresponding line types.
- Create the import data file. Data files consist of lines with a command followed by data.

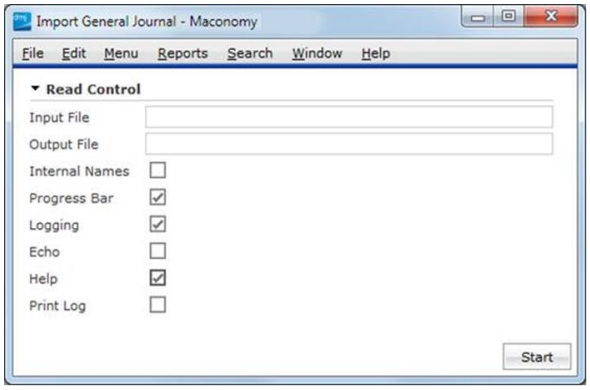
Activating Help

Select the Help field to produce a list of the line types available in the program. When you activate Help, no import is made. Instead, Maconomy creates a file with the line types available in the specified program. The line type names in the help file are the external or internal names, depending on whether you selected the Internal Names field (as described later in this document). The file is automatically placed in your home folder, and the name of the import program is used as the file name, with a .**txt** extension.

For more information about the use of each line type, see the section “Line Types” in this manual, and the description of the individual line types in the program-specific manual.

To activate Help, do the following:

1. Start the import program. Select **Menu » Programs**, then select the appropriate import program. The import program window is displayed.



2. Select the **Internal Names** check box, or leave it blank to use external names.
3. Select the **Help** check box, and click **Start**.

This creates a file in your home folder (for example, C:\Users\<UserName>).

Activate PrintFormat

Activate PrintFormat to produce a list of available fields for the corresponding line types. When a printformat line is processed during import, Maconomy creates a line in the log file with the names of corresponding fields. For example, if you import general journals, creating a line with the line type JOURNAL:PRINTFORMAT, the log file shows a line with the fields that are available for lines of the type JOURNAL:FORMAT.

A PrintFormat line is a line that only results in a line in the log file. No data in Maconomy is affected by a PrintFormat line, and the line only consists of the line type field. Any other information on the line is ignored.

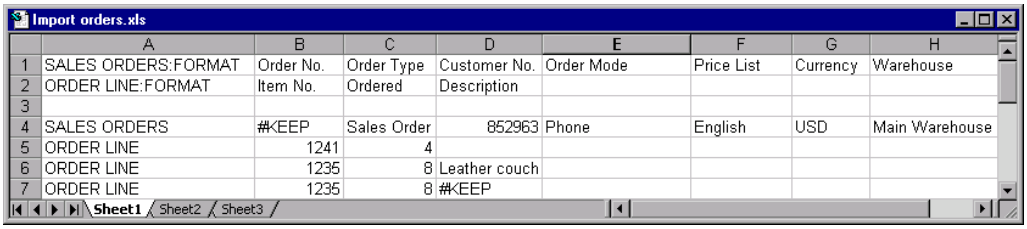
To activate PrintFormat, complete the following steps:

1. Open the help file from as described in the preceding section. Remove all lines except those including the word PRINTFORMAT, for example, Journal:PRINTFORMAT and save in a new file.
2. Start the import program. Select **Menu » Programs**, then the appropriate import program. The import program window is displayed.
3. In the **Input File** field, enter the name and folder of the file for which you want to print format. If you leave the field blank, you can select an input file after clicking **Start**.
4. Enter the path to an **Output File**. If you leave the field blank, the log file is placed in your home folder. The name will be that of the import program with the date appended.
5. Select the **Internal Names** check box, or leave it blank to use external names.
6. Click **Start.**

Create Data File

Import data files are tab-separated files consisting of lines with a command (line type) in the first field. For some commands, additional data is provided in the fields that follow the command. The file used to print format lines in the preceding section is in itself an import file with PrintFormat commands.

A typical import file could look like this:



	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Item No.	Ordered	Description				
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	1241	4					
6	ORDER LINE	1235	8	Leather couch				
7	ORDER LINE	1235	8	#KEEP				

The first two format lines identify which fields are used for importing. This sample shows SALES ORDERS and ORDER LINE, respectively. Next, data is provided for a sales order and a number of order lines according to the format specified in the FORMAT lines.

Save the import data file and import it using the same steps as in the previous section.

Reference Section

General Information on Standard Maconomy Import Programs

The functionality of Maconomy import programs follows a standard guideline, meaning that the programs used for importing different types of information generally work in a similar way. The import programs all offer standard functionality, which is explained in “Standard Line Types.” Standard functionality is defined as functionality that performs similar actions and is used in a similar way, regardless of the program. For instance, all standard import programs (with a few exceptions) can delete entries, and the method for deleting entries is the same for all import programs. For a definition of the concept of entries, see “Central Concepts in Import Programs”.

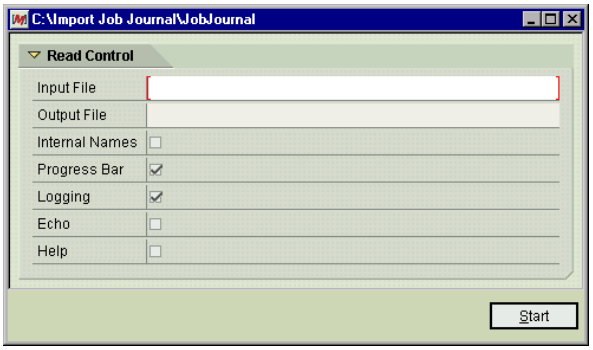
In addition to the standard functionality, some import programs offer a number of functions specific to the program in question. For instance, the Import Orders program allows the approval of sales orders, which is a function that is only relevant when working with sales orders. The functions that are specific to each program are described in the manual for the program in question. However, before reading about both the general functionality and the program-specific functionality, it is important that you understand the concepts defined in “Central Concepts in Import Programs”, because the remaining sections in this manual contain several references to these concepts.

Using the Program

This section describes how import programs are used.

Running Import Programs Manually

The import program is started by selecting the option "Programs…" in the Menu menu, and then selecting the appropriate import program. When you select the import program, Maconomy displays a window similar to the one in the following figure, where you can select the import file to be used and specify a name and folder for the log file that is created as a result of the import. The log file contains information about the results of the import as well as any errors that occur in the import file.



In the **Input File** field, you can enter the name and folder of the import file on the server. You should always specify the full path of the file, including the name of the folder in which the file is located. If you did not select **Print on Server** in the File menu, if you do not enter a file name or if the specified file does not exist on the server, Maconomy displays a dialog box where you can select the appropriate file. In the **Logging** field, you can enter the name and folder for the log file. If you do not provide a value for this field, Maconomy places the log file in the Maconomy folder, using the name of the import program followed by today's date as the file name.

Start the import by clicking **Start**. The dialog box remains open. If the import fails, for example, as a result of importing a pop-up value that does not exist in Maconomy, you therefore only need to edit the import file and click **Start** to make another attempt.

In a number of Maconomy's import programs, the window also contains the **Internal Names**, **Progress Bar**, **Logging**, **Echo**, and **Help** fields. The functionalities of these fields are described in the following sections.

Internal Names

This field determines whether the import file should contain internal or external field and line type names. If this field is not selected, the import file should contain external names, while internal names should be used if the field is selected. The internal names are the names that the fields and actions have in Maconomy's relations, meaning that they do not change from one version to another. When you are using internal names, the field names on format lines should be the internal names from the field lists, and line type names should be those specified as internal in the list of available line types. For more information about the concept of format lines, see "Format Lines." The concept of line types is described in "Line Types."

Progress Bar

In this field, you can specify whether Maconomy is to display a window during the import, displaying the number of lines imported, the number of errors encountered, and so on.

Logging

In this field, you can specify whether Maconomy is to continue processing the lines in the import file after encountering an error. If this field is selected, any errors in the import file are reported in the log file, and Maconomy continues processing the import file to see if there are other errors. If this field is not selected, Maconomy stops the import when the first error occurs, and the error message is shown on the screen. When an error is encountered, none of the information that was read since the last save point found in the import file is imported. Therefore, if you want to use the save point functionality, you should select this field; otherwise, the import will stop when an error is encountered. Because one of the purposes of using save points is to import all of the correct sections of the import file and to have all erroneous sections written to the log file, this field should be selected to ensure that the import continues after encountering errors.

Echo

In this field, you can specify whether the log file is to show what information was created and changed as a result of the current import. If this field is selected, each imported data line results in a line in the log file, showing the information that was imported to the fields in the format.

Help

If you select this field, no import is performed. Instead, Maconomy creates a file with the line type names that are available in the program in question. The line type names in the help file are the external or internal names, depending on whether you selected the **Internal Names** field. The file is automatically placed in the Maconomy folder, and the name of the import program is used as the file name. The file gets the extension ".txt." For more information about the use of each line type, see "Line Types" in this manual and the description of the individual line types in the program-specific manual.

Running Imports from Command Lines

You can run the import program by calling it from a command line in a Unix or Windows NT system. When doing so, the import is processed as normal, but you must specify the name and location of the import file to be used. You can also specify values for all of the fields in the import dialog described in “Running Import Programs Manually.” To type command lines, you must have a certain amount of knowledge of command-line syntax and the structure of your Maconomy installation. Understanding the following description therefore also requires basic knowledge in these areas.

When running import programs from command lines, you must first log in as the user Maconomy. The import program must be located on the server, meaning that on Unix systems, the file must be transferred through binary copy from the installation CD, whereas on Windows NT systems, it can simply be copied from the installation CD to the server.

Command lines that execute import programs are similar on Unix and Windows NT systems. The syntax for a command line to run an import program is as follows, where the information in brackets represents certain specific information that depends on your Maconomy installation:

```
[Maconomy Server Program] -S[Shortname] -x[Import Program]
```

The `[Maconomy Server Program]` should be the location and name of the Maconomy server program. The -S indicates that a shortname is specified next, and the [Shortname] should contain the shortname of the Maconomy database on which the import should be run. The -x tells the Maconomy server to execute the program that is specified next. The [Import Program] should be the location and file name of the import program. The path and file name should be specified in quotes. You can leave out the -S[Shortname] on the command line. If you do so, Maconomy prompts you for the shortname when the command line is executed. If you do not know the location of the Maconomy server program and the shortname, consult your Maconomy system administrator. The following is an example of a command line, in Unix and Windows NT respectively, which executes an import program for expense sheets in version 5.0.

Unix

```
macoracle.r.us_5_0 -Sus501 -x"Import ExpSheets 5.0.prm"
```

Windows NT

```
maconomyserver -imaconomyserver.us_5_0 -Sus501 -x"Import ExpSheets 5.0.prm"
```

This command activates the Maconomy server program and tells it to execute the "Import ExpSheets 5.0.prm" program on the database whose shortname is "us501". Because no path is specified for the import program, Maconomy assumes that the program is located in the current folder.

When the command line is executed (and, if you did not include it in the command line, you have specified a short name), Maconomy prompts you for any parameters to be changed in this import. The parameters correspond to the fields in the import dialog shown when the import program is activated manually, as described in “Running Import Programs Manually”. The parameters and their use are listed in “Parameters,” because this section only describes how to specify the parameters.

In the prompt, you can specify the name of a parameter and press Return. You are then prompted for a parameter value.

You can also enter $ for a complete list of the available parameters and their current settings.

The following example shows a prompt for parameter values and the specification of a parameter name and value. In this case, the parameter is INFILENAME, which is a parameter used to identify the location and name of the import file to be used.

```
Variable to change ($ to list all, <return> to continue, $$ to
   exit)INFILENAMEVARINFILENAMEVAR      : STRING             =

   , newvalue :Impfile.data
```

The first line is the prompt that is issued by the system. The second line is the parameter name specified by the user. On the third line, the first part is the prompt, and the text emphasized in bold is the parameter value (file name) specified by the user.

When you have specified all of the appropriate parameter values, press Return when prompted for a new parameter, and the import is performed.

Parameters

When you run import programs from command lines, the fields in the import dialog are referred to as parameters to which you assign a value for each import.

Parameters are specified through parameter names and parameter values. The parameter name indicates the parameter for which you want to specify a value, and the parameter value indicates the value to be applied to the parameter in question. For instance, the parameter name could be "InFileNameVar," and the parameter value could be "\Imports\Import Expense Sheets.txt." In this case, the parameter name indicates that the import file is now going to be specified, while the parameter value is the actual location and name of the import file. This corresponds to entering a file name in the **Inputfile** field in the import dialog when running an import manually.

Each parameter has a default value that is used if no other parameter value is specified.

The following list shows the names of the parameters that can be used, and what they are used for. The list also shows which values are allowed, and what the default value of each parameter is.

Parameter Name	Functionality	Default Value
InFileNameVar	Corresponds to the **Inputfile** field in the import dialog, as described in “Running Import Programs Manually.” Specify the name and location of the import file. It is mandatory to specify an input file. If you run the import from a command line, and you do not specify an input file (or if the specified input file does not exist), you are prompted for an input file.	Blank

Parameter Name	Functionality	Default Value
OutFileNameVar	Corresponds to the **Output Data File** field in the import dialog, as described in "Running Import Programs Manually." Specify the name and location of where to place the log file.	Blank
UseInternalNamesVar	Corresponds to the **Internal Names** field in the import dialog, as described in "Running Import Programs Manually." The value 1 ("true" in Workflow Spider scripts) corresponds to the **Internal Names** field being selected in the import dialog. The value 0 ("false" in Workflow Spider scripts) corresponds to the field not being selected in the import dialog.	0/"false"
UseProgressBarVar	Corresponds to the **Progress Bar** field in the import dialog, as described in "Running Import Programs Manually" The value 1 ("true" in Workflow Spider scripts) corresponds to the **Progress** field being selected in the import dialog. The value 0 ("false" in Workflow Spider scripts) corresponds to the field not being selected in the import dialog.	1/"true"
LoggingVar	Corresponds to the **Logging** field in the import dialog, as described in "Running Import Programs Manually." The value 1 ("true" in Workflow Spider scripts) corresponds to the **Logging** field being selected in the import dialog. The value 0 ("false" in Workflow Spider scripts) corresponds to the field not being selected in the import dialog.	1/"true"
EchoVar	Corresponds to the **Echo** field in the import dialog, as described in "Running Import Programs Manually." The value 1 ("true" in Workflow Spider scripts) corresponds to the **Echo** field being selected in the import dialog. The value 0 ("false" in Workflow Spider scripts) corresponds to the field not being selected in the import dialog.	0/"false"
PrintHelpVar	Corresponds to the **Help** field in the import dialog, as described in "Running Import Programs Manually." The value 1 ("true" in Workflow Spider scripts) corresponds to the **Help** field being selected in the import dialog. The value 0 ("false" in Workflow Spider scripts) corresponds to the field not being selected in the import dialog.	0/"false"

Parameter Name	Functionality	Default Value
BatchRunVar	With this parameter, you can specify whether Maconomy should issue a message when the import is complete. If a script should perform additional commands after the import, this parameter should be set to 1 ("true" in Workflow Spider scripts), because the script will otherwise be halted until you click the OK button in the message box.	0/"false"
APIInFileNameVar	This is a special parameter type. In addition to corresponding to the **Inputfile** field in the import dialog, as described in "Running Import Programs Manually," the use of this parameter indicates given values for the other parameters that can be used. When you have specified this parameter name, the other parameters for the program are automatically assigned the following values: ▪ UseInternalNamesVar: true UseProgressBarVar:= false ▪ LoggingVar: false ▪ EchoVar: true BatchRunVar: true As the parameter value for this parameter, specify the name and location of the import file. It is mandatory to specify an input file. When you have used this parameter, you cannot use the **OutFileNameVar** parameter. Instead, you must use the **APIOutFile-NameVar** parameter to specify a log file location.	Blank
APIOutFileNameVar	This parameter type is only allowed when the input file name was specified using the **APIInFileName** parameter. In those cases, you must use this parameter to specify a location and name for the log file.	Blank

Using Parameter Files in Command Lines

Instead of specifying the relevant parameters one by one as described in “Running Imports from Command Lines”, you can create a parameter file that includes all of the parameters to be used. A parameter file is simply a text file with the specification of parameters and values to be used in an import. When you use a parameter file, you do not need to manually specify anything after the command line is run. This allows you to run imports in batches where a standard set of parameter values are used in a completely automatic procedure.

The syntax used for identifying a parameter file corresponds to the one used when specifying the parameters manually, one by one, but you add the location and name of the parameter file as an attribute to the file name of the import program. Thus, the command line should look as follows:

```
[Maconomy Server Program] -S[Shortname] -x[Import Program]
```

```
<[Parameter File]
```

Note that a < must be specified before the path to the parameter file. The following is an example of a command line that executes an import program for expense sheets, using the parameter values specified in the "Import.parms" file.

```
macoracle.r.us_5_0 -Sus501 -x"Import ExpSheets 5.0.prm"
  <"Import.parms"
```

The parameter file must also comply with a number of syntax rules that are described in the following.

If the command line does not include the shortname, the parameter file must begin with a line with the shortname. However, if the shortname is included in the command line, only the following information should be included in the parameter file.

For each parameter for which you want to specify a value, create a separate line on which you specify the name of the parameter followed by a "=" and the value to be assigned. After the last parameter, insert a blank line to indicate that no more parameters are to be specified, and the import should continue.

The following is an example of the lines in a parameter file. It is assumed that the shortname was specified on the command line:

```
InFileNameVar= "ExpenseSheets.data"
OutFileNameVar="ExpenseSheets.log"
BatchRun=1
<Blank line>
```

Running Imports from Workflow Spider Scripts

You can run imports from Workflow Spider scripts. The command syntax is as described for running reports in the Maconomy Workflow Spider reference manual. When you use Workflow Spider scripts, both the program file and the import file must be located on the Maconomy server.

In Workflow Spider reports, the report file name is the name of the import program's program file. The `InputVariables` attribute should list the names of the parameters for which you are going to specify values. If you list several parameter names, separate them with commas and specify them in the order in which you are going to specify their values in the `InputValues` attribute. The `InputValues` attribute should list the values for the parameters specified in the `InputVariables` attribute. If several parameters are used, specify the values in the order in which the parameter names were listed in the `InputVariables` attribute, and separate the values with commas. For variables whose value can be "true" or "false", the value must be specified in quotes.

The following example illustrates a script fragment that runs an import of expense sheets:

```
<ExecuteReport ReportFileName="Import Expense Sheets
    5.2.prm"InputVariables=(InFieNameVar,UseProgressBarVar,UseInternalNameVar,B
    atchRunVar)InputValues=("\import data\Expense Sheets.txt", "false", "true",
    "true")>
```

In this example, the InputVariable attribute specifies four different parameter names, and the order in which the values for these parameters are specified in the `InputVariables` attribute. The `InputValues` attribute contains the values to be used for each of these parameters. The input file is thus "\import data\Expense Sheets.txt" on the Maconomy server. The value for `UseProgressBar` is "false"; that is, no progress bar should be shown. The value for `UseInternalName` is "true"; that is, internal names are used, and the value for `BatchRun` is "true," meaning that no message is shown after the import.

For the parameters not mentioned in the script, Maconomy uses the default values.

As always in Workflow Spider scripts, the parameter values need not be specified explicitly. For instance, the input file name could consist of a fixed path combined with the name of a file created as a result of a command earlier in the script. For instance, the file name could be the number of the job for which the script is run. In this case, the input file name could be specified as follows:

```
InputVariables=(InFileNameVar)
InputValues=("C:\import data\"+JobNumber)>
```

Central Concepts in Import Programs

This section contains a definition of a number of concepts used in this manual. This manual contains several references to these concepts, and it is therefore important that you understand the concepts before you read the rest of the manual.

Import File

The import file is the file that contains the information to be imported. Each import requires the input from one import file. The import file must be a standard tab-separated text file. The recommended way of working with an import file is by using a spreadsheet, because a spreadsheet program provides an effective overview of the lines and fields in the file, and it also offers a number of editing tools that make it easier to change and move information in the file. However, you should be aware of the auto-correct features of certain spreadsheet programs, because some information may be auto-corrected so that, for example, vital spaces between figures and %-characters are removed, resulting in an error. After you have completed the import file, save it as a tab-separated text file.

Depending on whether you are importing data from a Macintosh client or a Windows client, the following rules apply to the file format of the import file:

If the import is performed from a Macintosh client, the import file must be a standard tab-separated file in the Macintosh extended ASCII characters. The entries are separated by cr (Carriage Return). The ASCII values for cr and tab are 13 and 9, respectively.

If the import is performed from a Windows client, the import file must be a standard tab-separated file in Windows characters. Line shift in a Windows text file is represented by cr (carriage return) and nl (new line), which have the ASCII values 13 and 10, respectively. The ASCII value for tab is 9.

Import Lines

An import line is simply a line in an import file. In a spreadsheet, an import line corresponds to a row.

Entries

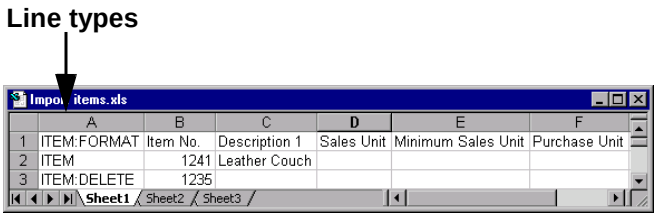
An entry is a record in a relation in Maconomy's database. For instance, a job shown in the Jobs workspace is an entry in the Jobs relation in Maconomy's database. A tab in a workspace in Maconomy shows one entry at a time, while the sub-tab of the workspace shows an entry on each line. The following is an example of data in the customer relation in Maconomy. The first row contains the names of the fields that make up the relation, while each of the remaining rows represents an entry in the relation.

Customer No.	Name 1	Name 2	…	Changed by
10995	Hugo Rune	324 Ealing Road		Administrator
10997	Colin Danbury	110 Johnson Avenue		Administrator

Customer No.	Name 1	Name 2	…	Changed by
10998	Legalease	28-33 Cato Street		Administrator

Line Types

Each import line is of a certain type. You specify the line type in the first field (column A) of each import line. The line type specifies the contents of the import line. For instance, the line type can specify that the line in question contains information to create/update an entry, or that the line should perform a given action from the Action menu in Maconomy.



Although the line types that are available in each program vary, a number of the line types work in a similar way, meaning that the functionality of the line types is the same, but the lines affect different types of entry. For example, all standard import programs (with very few exceptions) contain a line type for deleting entries. These line types work in exactly the same way, but they are named differently according to the type of entry to be deleted, for example, ITEM:DELETE is available in the Import Items program, and SALES ORDERS:DELETE is available in the Import Orders program. Line types whose functionality is common to all programs are called standard line types. The general functionality of each standard line type is described in detail in “Standard Line Types,” and each program-specific manual contains a list of the line types that are available in the program in question. You can thus check the available line types in the manual for any import program and read about their functionality in “Standard Line Types” in this manual.

Each line type pertains to a given relation in the Maconomy database, meaning that the information that is imported is not necessarily limited to one Maconomy window. For instance, in the Import Purchase Orders program, you can use the line type PURCHASEORDERLINE, which specifies that the import line contains information to be imported to the Maconomy PurchaseOrderLine relation. This relation contains information about both the number ordered on a purchase order line and the number received. Therefore, data on lines of this type can be related to both the sub-tab of the Purchase Orders workspace and the Item Receipt workspace.

Keys and Key Fields

A key field is a field that is part of the key to a given entry in Maconomy’s database. A key consists of one or several key fields whose combined values identify a given entry in Maconomy’s database. No two keys in a database relation can be identical. Therefore, when you need to identify a given entry, such as an item or an order, you do it by specifying the key. For example, the key to a sales order is the order number, and no two sales orders can have the same number. If you want to update a sales order using an import program, you therefore identify the order to be updated by specifying the order number of the order in question.

Of course, when an entry is created, it must be assigned key values to make it unique. This is sometimes done manually, and sometimes automatically by Maconomy, depending on the setup of your system and the workspace into which information is being imported. For example, if you are creating a requisition, you can only specify the requisition number manually if the **Manual Requisition Numbers** system parameter is selected. If this parameter is not selected, you must let Maconomy assign a number to new requisitions. You let Maconomy assign a value to a field

by entering #KEEP in the field in question. For more information about standard values, see “The #KEEP Command.”

The key fields are specific to each Maconomy database relation. Please refer to MDoc for information about key fields for each relation. Some relations have a number of alternative sets of key fields. In that case, the available sets of key fields are listed in the help for the import program. To select a set different from the primary key, use a KEYFIELDS line from the help.

Keys in Table Parts

The key in relations that represent lines in the sub-tab of a given workspace, for instance order lines or inventory change lines, consists of the key to the tab to which the line is assigned as well as a line number that represents the location of the line in the sub-tab. In a few cases, the line number is not part of the key. Instead, another type of value is used in the key. For instance, the key to a line in the sub-tab of the Item Information Card workspace consists of an item number and a warehouse because a given warehouse can only be represented on one line for each item.

States

Imported data is processed in the same way as manually entered data. As in Maconomy, there are two states for data entry: Creating and Updating. Creating is the state of an entry being created, meaning that certain fields can/must be completed, while others are closed for data entry.

In Maconomy, an entry is in the Creating state from the time when you create an entry by using the Index menu, š+N/CTRL+N or activating a new sub-tab line, until you press Enter.

Updating is the state that an entry has after being created. In Maconomy, an entry is in the Updating state when you have pressed Enter after creating an entry or when you are editing an existing entry. For instance, in the Updating state of an item, you cannot change the item number (as well as a number of other fields).

Each field can be either open or closed in each of the two states. Open means that a value can be entered, while closed means that you cannot enter or change the value in the field.

It is possible for a field to be closed in both states. For instance, the fields in the User island are always closed. When you import information, Maconomy allows values in fields that are closed both in the Creating and Updating states. These fields are ignored during the import, but can be included in the format, thus allowing you to export data from Maconomy, edit the information, and reimport it without having to remove the columns containing closed fields.

In the field lists from the format lines in the help output, the Open column shows the states in which the individual fields are open. Open in the Creating state is marked with a C, while open in the Updating state is marked with a U. Open in both states is marked with a check mark, Ð, while the column shows a ÷ for the fields that are closed in both the Creating and Updating states. In most cases, you can also find out if a field is open or closed in each state by creating an entry in the window in question and observing which fields are open before and after you press Enter.

Mandatory Fields

In the states where a field is open, the field can be either mandatory or optional. Refer to MDoc for information on which fields are mandatory and whether the field is open for input when creating and/or changing records. Mandatory when open means that if a field is, for example, mandatory but only open in the Creating state, the rules for data in mandatory fields only apply when entries are created, while the contents of the field are ignored in the Updating state where the field is closed. The rules for data in mandatory fields are as follows:

- Mandatory means that fields of the types String, Date, Time, and Popup cannot be blank, while Integer, Real, and Amount fields cannot be zero. Thus, the import file must not contain these values in mandatory fields. Fields that are mandatory and open in the

Creating state must be included in the format when creating entries. In the Updating state, mandatory fields that are not included in the format keep their existing values.

- Mandatory does not always mean that you must manually enter a value. In some cases, you can let Maconomy assign the standard value or keep the existing value by specifying the value #KEEP in the field in question. Whether a value must be entered manually or not depends on the exact functionality of the window into which you are importing information. It is therefore not possible to define a set of rules that apply to all mandatory fields. However, as a rule, you can avoid many problems by always including mandatory fields in the format and always completing these fields.

Standard Import Functionality

This section describes the features available in import programs, and how to use them.

Information Processing

This section describes how information is processed by the import program. Part of the standard functionality of import programs is the fact that all information that is imported is treated as if it were entered manually in a Maconomy window. This means, for example, that the type of information (dates, text, amounts, and so on) in a given field that is imported must comply with the type of information allowed in the corresponding field in Maconomy. In addition, the rules regarding valid data in Maconomy also apply to import programs, meaning that, for instance, any pop-up values, customers, or items specified in an imported field must exist in Maconomy. In some cases, a given field may only contain a value if a number of conditions are met. For instance, in the General Journal workspace, you can only enter a value in the **Vendor Inv. No.** field if the **GRP** field contains the value "P," and you have stated a vendor number. This type of condition also applies to imported data. It is therefore important to know the exact functionality of the window or windows into which you are importing information before you set up your import file.

For more information about the exact functionality of Maconomy fields and windows, see the Maconomy Reference Manual.

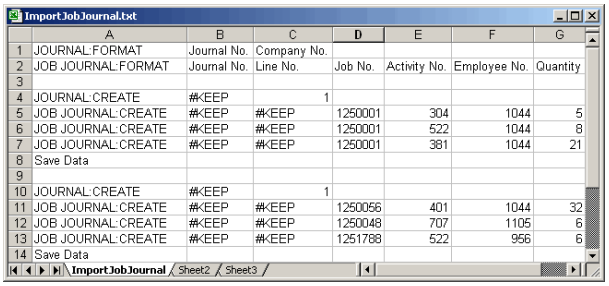
Data Commitment to Database and Errors During Import

When you import information, the entire import is performed as one database transaction. Consequently, an error in any single line, such as a pop-up value that is not recognized by Maconomy, means that none of the information in the import file is imported. If the import contains errors, you can find the error messages in the log file created by the import program.

However, most import programs allow you to insert “save points” in the import file. Each time a save point is reached without encountering errors, Maconomy writes the information imported since the last save point to the database. If errors are encountered between two save points (or between the beginning of the file and the first save point or the last save point and the end of the file), the information in that section is rolled back (cancelled), meaning that the data in that section is not imported. Instead, all of the import lines between the two points are written to the log file. After the import is completed, all data in successfully imported sections thus has been saved in the database, while the log file contains all of the erroneous sections, allowing you to copy those sections to a new import file, correct the data that caused the errors, and run a new import with the corrected lines. Please note that this functionality requires that the **Logging** field is selected in the import dialog (see the description of the **Logging** field), because the import is otherwise stopped the first time that an error is encountered (however, data saved by means of save points before the error is encountered is, of course, saved in the database).

To insert a save point, create a line at the relevant point in the import file and enter “Save Data” in the first field if external names are used in the import and “Commit” if internal names are used.

The following figure provides an example.



ImportJobJournal.txt

	A	B	C	D	E	F	G
1	JOURNAL:FORMAT	Journal No.	Company No.				
2	JOB JOURNAL:FORMAT	Journal No.	Line No.	Job No.	Activity No.	Employee No.	Quantity
3							
4	JOURNAL:CREATE	#KEEP	1				
5	JOB JOURNAL:CREATE	#KEEP	#KEEP	1250001	304	1044	5
6	JOB JOURNAL:CREATE	#KEEP	#KEEP	1250001	522	1044	8
7	JOB JOURNAL:CREATE	#KEEP	#KEEP	1250001	381	1044	21
8	Save Data						
9							
10	JOURNAL:CREATE	#KEEP	1				
11	JOB JOURNAL:CREATE	#KEEP	#KEEP	1250056	401	1044	32
12	JOB JOURNAL:CREATE	#KEEP	#KEEP	1250048	707	1105	6
13	JOB JOURNAL:CREATE	#KEEP	#KEEP	1251788	522	956	6
14	Save Data						

ImportJobJournal / Sheet2 / Sheet3

Inserting frequent save points increases the performance of the import program because the program does not have to manage a large amount of data at a time. In addition, time is saved, because correct import data does not have to be reimported due to errors that occur later in the import file.

However, save points should only be inserted after whole sections of related import data (for instance, after all of the lines that pertain to a given sales order), not after each line. Consider a case where you want to run an import that creates a number of sales orders, each of which contains a number of order lines. If you place a save point after each line and an error occurs on one of the lines with order line information, the erroneous line is written to the log file, while the remaining information is successfully written to the database. However, the line that is written to the log file now stands out of context, and you may be unable to tell which order the line belongs to.

It should be noted that using save points slightly increases the risk of database deadlock, because the relations that are used in the import are unlocked each time date is committed. This makes it possible, for example, for other import programs that run at the same time to lock those relations, thus preventing the import from continuing.

As mentioned, this functionality is available in most standard import programs from Maconomy. To find out whether the functionality is available in a given program, use the **Help** field in the import program dialog, and see if the help file contains a line with the text “Save Data” or “Commit,” depending on your choice of external or internal names.

Field Types

This section contains a description of the values allowed in different types of fields in the import file. Each field is of a certain type that defines the values that are allowed in the field in question. For instance, some fields can contain text, while others can only contain figures. In your import file, it is therefore important that the contents of each field are in compliance with the field type of the field in question. See MDoc for information on the type of each field. The field types are described in the following table.

Type	Example	Description
String	Leather Couch	This field type is used for text strings. The field can contain a maximum of 255 characters, and both letters, figures, and special characters can be used.
Integer	999	This field type can only contain figures. No decimals are allowed. A blank value that is imported results in the number 0.
Real	999,999	This field type can only contain figures. The field type allows decimals, and the comma is used as the

Type	Example	Description
		separator, for example, 372,1. The maximum number of decimals that is allowed is the same as the chosen number of decimals in the Preferences workspace (in the Edit menu) in Maconomy. If you import a real figure that has fewer decimals than set up in the Preferences workspace (or none at all), Maconomy adds the remaining decimals automatically. A blank value that is imported results in the number 0,00 (and the relevant number of decimals).
Amount	99,99	This field type is used for amounts. The comma is used as the separator between the main monetary unit and the secondary monetary unit. A maximum of two decimals is allowed, for example, 125,00. A blank value that is imported results in the amount of 0,00.
Date	07.01.02	This field type is used for dates, and follows the format day, month, year. A dot is used as the separator between day, month, and year, for example, 07.01.99. A blank value that is imported results in a blank field in Maconomy.
Time	13:45:00	This field type is used for time. The colon is used as the separator, for example, 13:45:00. No blanks are allowed before or after the value. A blank value that is imported results in a blank field in Maconomy.
Checkbox	1	This field type represents check boxes in Maconomy. The values allowed are 0 and 1. 0 stands for an unselected check box, while 1 stands for a selected check box. A blank value that is imported results in an unselected check box.
Popup	sssss	This field type is used for options in pop-up fields. You specify a pop-up value by the name of the option, and the option must already exist in Maconomy. A blank value that is imported means that no option is selected in the pop-up field in question. Note that a blank value is only allowed if the pop-up field in question can be left blank in Maconomy.

Standard Line Types

This section describes the standard line types in Maconomy import programs. Because the standard line types function in a similar way in all import programs, the standard line types are described on a general level. A number of import programs allow you to choose between the use of internal or external names. The internal names are the names that are used in Maconomy's relations, while the external names correspond to the field headers and action names in the Maconomy windows. In the program-specific manuals for the programs where this feature is available, the list of available line types shows both the external and internal names of each line type. All lines in the import file should thus contain either the internal or external names of both fields and line types in accordance with the choice made when the import is started.

Standard line types can be divided into five groups: format lines, data lines, action lines, format return lines, and comment lines. The line types within each group are described in the following section. Note that some of the line type names contain the term <OBJECT>. The object is the type of information to which the line type pertains. When you set up your import file, replace the term <OBJECT> with the name of the relation that the line concerns. For instance, when importing items, replace the line type <OBJECT>:FORMAT with ITEM:FORMAT. The objects that can be handled vary from program to program. Some programs can even handle several objects. For instance, Import Orders can handle the objects SALES ORDERS and ORDER LINE.

Format Lines

Line type name: `<OBJECT>:FORMAT`

Examples: `ITEM:FORMAT, ORDER LINE:FORMAT, JOURNAL:FORMAT`

On a format line, you can specify the names of the fields to which you want to import data that pertains to the object that is specified in the name of the format line type. The order of the field names specified on a format line also determines the order in which data should be entered on subsequent data lines that pertain to the object in question. A format line does thus not result in the creation/update of any entries, nor does it perform any actions.

The following example shows an import file for the import of orders.

Import orders.xls

	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Item No.	Ordered	Pick	Description	Extra Text 1		
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	1241	4	4	Leather couch	Special order		
6	ORDER LINE	1235	8	8	King size bed, oak			
7	ORDER LINE	1235	15	15	King size bed, larch			
8	SALES ORDERS:CALCULATE QUANTITY DISCOUNT	200009						
9	ORDER LINE:PRINTFORMAT							
10		Customers abroad below this line						

Sheet1 Sheet2 Sheet3

- Row 1: Format line for the object SALES ORDERS.
- Rows 4 and 8: Lines with object SALES ORDERS must comply with the nearest preceding line of the type SALES ORDERS:FORMAT.

In the Import Orders import program, you can import data for two objects: SALES ORDERS and ORDER LINE. Therefore, you must set up a format for each object, specifying the fields to which you want to import data. In the preceding example, the SALES ORDERS:FORMAT line type in line 1 specifies that whenever the SALES ORDERS object is used on subsequent lines, column B should contain an order number, column C should contain an order type, column D should contain a customer number, and so on. The ORDER LINE:FORMAT line type on line 2 specifies the order of fields for the ORDER LINE object in the same way.

On lines 4 and 8, you can see from the line type names that the lines pertain to the SALES ORDERS object. The data on these lines therefore must be presented in the order that is specified by the format for the SALES ORDERS object, that is, line 1. Similarly, lines 5-7 pertain to the ORDER LINE object, and therefore must comply with the format on line 2.

The field names available are specific to each object, and the format line types available are specific to each import program. See “Getting Started” for information about how to print the available format lines and the field supported by each object. The field names on format lines must match the names in the field list with regards to spelling, spaces, punctuation, and so on; otherwise, the import will fail.

A number of import programs allow you to choose between the use of internal or external names. The internal names are the names that are used in Maconomy’s relations, while the external names correspond to the field headers in the Maconomy windows. In the program-specific manuals for the programs where this feature is available, the field lists show both the external and

internal names of each field. All format lines in the import file should thus contain either internal or external field names in accordance with the choice made when the import is started.

When you set up your import file, you must specify a format line before you can specify any data and action lines; otherwise, the import program cannot tell which column pertains to which field.

A format line applies until another format line for the same object occurs in the import file.

The format for a given object does not need to include all of the available fields. If you leave fields out of the format, the fields that are not specified on the format line are treated as if they contained the command #KEEP on all subsequent lines with that object. The effects of the command #KEEP are described in the “The #KEEP Command.” Note that on data lines for a given object, you need not enter information in all of the fields in the format. For instance, in preceding figure, the **Extra Text 1** field (column F) has been left blank on lines 6 and 7. For more information about the effects of leaving a field blank, see “Leaving Fields Blank.”

Data Lines

Line type name: `<OBJECT>, <OBJECT>:CREATE, <OBJECT>:CHANGE`

Examples: `ITEM, ORDER LINE, JOURNAL, ITEM:CREATE, ITEM:CHANGE`

A data line contains the data values to be imported to the fields specified in the nearest format for the object in question. Thus, a data line can result in the creation and/or update of an entry in Maconomy’s database. There are three different forms of data lines. By using different forms of data lines, you can specify whether the information on the individual line should result in the creation of an entry, the update of an existing entry, or whether Maconomy should determine whether an entry is to be created or updated, based on the values in the key fields of the line. However, in some programs, you can only let Maconomy determine whether an entry should be created or an existing one should be updated.

The value in each field on a data line must comply with the data that is allowed in the field type of the field in question. For instance, the **Pick** field in the previous example is of the type Integer, meaning that no letters are allowed in this field. You can find the field type of each field in the field lists in the program-specific manuals, while the exact meaning of each field type is described in “Standard Import Functionality” in this manual.

The concept of key fields is explained in “Central Concepts in Import Programs.”

The following describes the three forms of data lines:

- **Data lines for creating entries**

  Line type names: `<OBJECT>:CREATE`

  Examples: `ITEM:CREATE, INVOICELINE:CREATE`

  In most import programs, you can specify that a data line should result in the creation of an entry in the database. The line type name of such a line consists of the name of the object for which the line contains data, followed by a colon and the text "CREATE."

  When such a line type is used, an entry is created in Maconomy with the data specified on the line. However, only fields that are open in the Creating state are imported, because any values in other fields are ignored. This corresponds to selecting "New <OBJECT>" in the Index menu in Maconomy, entering information in the fields that are open for data entry, and pressing Enter.

  When you use this form of data line type, you ensure that existing entries are not updated by mistake. If Maconomy’s database already contains an entry with the key values specified on the data line, the import fails.

- **Data lines for updating existing entries**

Line type names: `<OBJECT>:CHANGE`

Examples: `ITEM:CHANGE, INVOICELINE:CHANGE`

In most import programs, you can specify that a data line should result in the update of an existing entry in the database. The line type name of such a line consists of the name of the object for which the line contains data, followed by a colon and the text "CHANGE."

When such a line type is used, the specified entry is updated with the data that is specified on the line. However, only fields that are open in the Updating state are imported, because any values in other fields are ignored. This corresponds to browsing to a given existing entry, changing some information, and pressing Enter.

When you use this form of data line type, you ensure that no new, undesired entries are created by mistake. The entry to be updated is identified from the values in the key fields on the data line in question. If Maconomy's database does not contain an entry with the key values specified on the data line, the import fails.

- **Automatic data lines**

Line type names: `<OBJECT>`

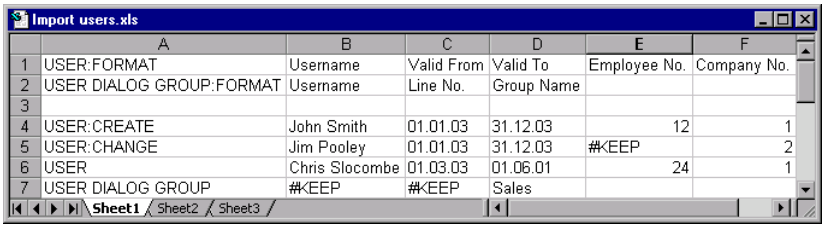
Examples: `ITEM,INVOICELINE`

In some cases, you want to let Maconomy determine whether an entry should be created or an existing entry should be updated. For this purpose, you can use automatic data lines. The line type name of a data line consists of the name of the object for which the line contains data. When the import is run, Maconomy checks whether the database already contains an entry with the key values specified in the key fields on the data line.

If the database already contains an entry with the key values on the data line, Maconomy updates the entry in question with the information on the line, just as if you had used the line type <OBJECT>:CHANGE.

If no such entry exists in the database, Maconomy creates the entry with the information on the line, and Maconomy processes the line as if you imported the same line twice, namely first with the line type <OBJECT>:CREATE and then with the line type <OBJECT>:CHANGE. Although some mandatory fields are only open in the Updating state, they must also contain a value when you create entries using automatic data lines. This is due to the fact that, as mentioned, the entry in question is first created and then updated. Therefore, if you are creating entries with fields that are mandatory and open in the Updating state, you must enter an explicit value or the #KEEP command in these fields; otherwise, Maconomy first creates the entry without problems, but afterwards attempts to update the mandatory fields with the value blank, resulting in an error.

The following example shows an import file for the import of users.



There are four data lines in this example: lines 4-7. Lines 4, 5, and 6 contain the object USER, meaning that the information should be entered in the order specified in the format for the USER object, that is, line 1. Therefore, column B contains a user name, column C contains a starting date for the user's validity, and so on. The information on line 7 pertains to the USER DIALOG GROUP object, meaning that the information should be entered in the

order specified in the format for the USER DIALOG GROUP object, that is, line 2. For more information about the command #KEEP, see “The #KEEP Command.”

On line 4, the line type is USER:CREATE. As a result, Maconomy checks that the user who is specified on the line does not already exist in Maconomy. Next, the new user is created. Maconomy only imports values from the fields open in the Creation state, ignoring any other information. If a user with the specified name already exists in Maconomy, the import will fail.

On line 5, the line type is USER:CHANGE. As a result, Maconomy checks that the user specified on the line has, in fact, already been created in Maconomy. Next, the user in question is updated. Maconomy only imports values from the fields that are open in the Updating state, ignoring any other information. If no users with the specified name exist in Maconomy, the import fails.

On line 6, the line type is USER. As neither :CREATE nor :CHANGE has been specified, this means that Maconomy checks whether the import should trigger the creation of a user or update an existing user. Maconomy uses the value in the key field (in this case the user name) to determine whether the user exists. If the user exists, the user information is updated with the information on the line. If the user does not exist, the information on the line is imported to a new Maconomy user. In this process, Maconomy first imports the fields that are open in the Creating state, and then updates the new user with the fields that are open in the Updating state.

Action Lines

Line type name: `<OBJECT>:ACTION`

Examples: `ITEM:DELETE, SALES ORDERS:APPROVE ORDER, SALES ORDERS:PRINT INVOICE`

An action line performs an action that corresponds to the one that is performed when you select the action from the Index or Action menu in a given window in Maconomy. You must specify the entry on which the action is to be performed, such as the order to be approved, the item to be deleted, and so on. You specify the entry by entering the key of the entry in question. For instance, the key to a specific order is the order number. When you specify the SALES ORDERS:APPROVE ORDER line type and enter the relevant order number in the **Order Number** field, the import program performs the **Approve Order** action on the order in question, just as if you had selected the action from the Action menu while the order was shown in the Sales Orders workspace. For a description of the concept of keys and key fields, see “Central Concepts in Import Programs.”

Action lines must comply with the nearest preceding format for the object in question. This means that the values in the key fields must be specified in the correct columns as defined in the format for that object. On action lines, any information in addition to the values in the key fields is ignored. You therefore only need to enter the relevant values in the key fields and leave any remaining fields blank on the action line. Note that the remaining fields on the action line do not have to be blank. Each remaining field can contain a value, as long as that value complies with the field type of the field in question. Returning to the example used in the illustration of format lines (shown in the following figure), you can see that the import file contains one action line, line 8.

Import orders.xls

	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Item No.	Ordered	Pick	Description	Extra Text 1		
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	1241	4	4	Leather couch	Special order		
6	ORDER LINE	1235	8	8	King size bed, oak			
7	ORDER LINE	1235	15	15	King size bed, larch			
8	SALES ORDERS:CALCULATE QUANTITY DISCOUNT	200009						
9	ORDER LINE:PRINTFORMAT							
10		Customers abroad below this line						

Sheet1 Sheet2 Sheet3

The SALES ORDERS:CALCULATE QUANTITY DISCOUNT line type indicates that the object is SALES ORDERS, and the action to be performed is the calculation of quantity discount. As mentioned, the key to an order is the order number. Therefore, the key to the order for which the discount is to be calculated should be specified in the order number column defined by the format for the SALES ORDERS object, that is, column B.

Format Return Lines

Line type name: `<OBJECT>:PRINTFORMAT`

Examples: `ITEM:PRINTFORMAT, ORDER LINE:PRINTFORMAT`

A format return line is a line that only results in a line in the log file. No data in Maconomy is affected by a format return line, and the line consists of only the line type field, meaning that the line only contains a value in column A. Any other information on the line is ignored. When a format line is processed during import, Maconomy creates a line in the log file that contains the names of the fields that are included in the nearest preceding format line that pertains to the object in question. This means that if, for instance, you are importing general journals, and you create a line with the JOURNAL:PRINTFORMAT line type, the log file will show a line with the fields included in the nearest preceding line of the JOURNAL:FORMAT type. If a format return line occurs before any format lines for the object in question have occurred in the import file, for instance because the format return line is the first line in the file, the line in the log file will show all of the fields that can be used on format lines for that object.

Line 9 in the following example is a format return line. It contains only the line type name.

Import orders.xls

	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Item No.	Ordered	Pick	Description	Extra Text 1		
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	1241	4	4	Leather couch	Special order		
6	ORDER LINE	1235	8	8	King size bed, oak			
7	ORDER LINE	1235	15	15	King size bed, larch			
8	SALES ORDERS:CALCULATE QUANTITY DISCOUNT	200009						
9	ORDER LINE:PRINTFORMAT							
10		Customers abroad below this line						

Sheet1 Sheet2 Sheet3

The ORDER LINE:PRINTFORMAT line type name indicates that the object is ORDER LINE. Therefore, the log file will show a line with the names of the fields included in the nearest above line of the ORDER LINE:FORMAT type, that is, line 2.

A number of import programs allow you to choose between the use of internal or external names. In these programs, Maconomy prints either the internal or external names in accordance with the choice made when the import was started.

Parameter Lines

Line type name: `<SPECIFICATION>PARAMETER SET:UPDATE`

Examples: `INVOICINGONACCOUNTPARAMETERSET:UPDATE, PARAMETER SET:UPDATE`

In some programs, you can use parameter lines. A parameter line is a line on which you can specify certain information that is not saved in the Maconomy database, but that is used in connection with actions that are performed in the import. An example of the use of parameter lines is the import of subscription orders where a parameter line allows you to enter information about the amounts to be invoiced on account when the next line of the SUBSCRIPTION ORDER:PRINT INVOICE ON ACCOUNT type is imported. Another example is the import of general journals where you can use a parameter line to specify whether journals posted in the import should be printed. In connection with parameter lines, an action can either be an action that is performed through an action line or the creation/change of information by means of a data line.

Information on a parameter line is specified in one or several fields, each of which has its own functionality. In the parameter set for invoicing subscriptions on account, you can, for example, specify in one field how much is to be invoiced for the subscriptions themselves, while you specify in another field how much is to be invoiced for tax.

As mentioned, the information in these fields is not saved in the Maconomy database, and it therefore cannot be displayed in the Maconomy system after the import. The parameter information is only used in the current import.

The value in each field on a parameter line applies until it is changed by importing another value in the same parameter field. This means that if, for example, in the import of general journals you specify on a parameter line that journals posted in the import should not be printed, this parameter setting will apply to all subsequent postings in the import until another parameter line occurs that contains another value in the same parameter field.

Note, however, that some actions reset the parameters, making it necessary to specify a new parameter line with the relevant settings after each action line.

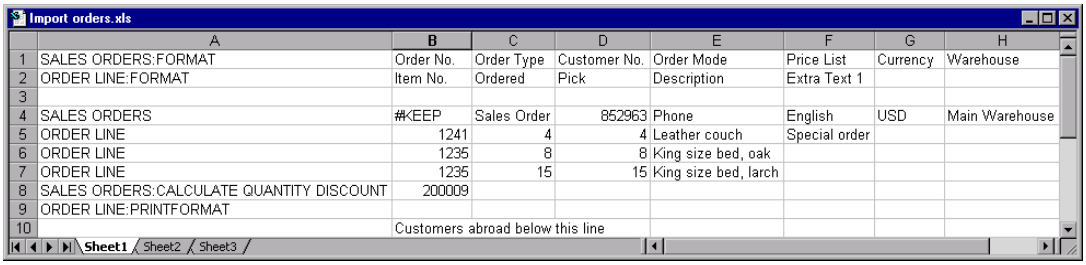
Each field on a parameter line has a default value that applies until it is changed on a parameter line of the type in question. If no parameter lines are specified in the import file, the standard value applies throughout the entire import.

Some parameter sets contain only one parameter field, while others contain a number of fields. Therefore, you must always specify a format line for parameter sets before an actual parameter line can be imported, regardless of whether the parameter set consists of only one field. Format lines for parameter sets are specified in the same way as normal format lines. The name of each available parameter set format line in a program is shown in the program-specific manual, and the "Field Lists" section in the program-specific manual contains a list of the fields that are available in each parameter set format. The field lists also show the default value of each parameter field.

Comment Lines

A comment line contains no data, formats, or actions. A comment line is created by leaving the first field (column A) blank. You can use comment lines for headings and notes. Comment lines are ignored during the import.

Lines 3 and 10 in the following example are comment lines, which is indicated by the fact that column A is blank. The text in column B on line 10 is just a note, and the line has no effect whatsoever on the import.



Import orders.xls

	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Item No.	Ordered	Pick	Description	Extra Text 1		
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	1241	4	4	Leather couch	Special order		
6	ORDER LINE	1235	8	8	King size bed, oak			
7	ORDER LINE	1235	15	15	King size bed, larch			
8	SALES ORDERS:CALCULATE QUANTITY DISCOUNT	200009						
9	ORDER LINE:PRINTFORMAT							
10		Customers abroad below this line						

Sheet1 Sheet2 Sheet3

The #KEEP Command

The #KEEP command is a command that allows you to let Maconomy assign a non-explicit value to a field—that is, the existing value from an entry being updated, the value from a previously imported entry, or a standard value. In general, the #KEEP command means that the field is treated in the same way as when you enter information in a given window but do not touch the field in question. The #KEEP command is used by entering "#KEEP" in a given field. When a field contains the #KEEP command, Maconomy processes the field as described in the following, depending on whether you are creating or updating an entry:

Using the #KEEP Command when Creating Entries

The following table shows how Maconomy assigns values to fields that contain the #KEEP command when creating entries (regardless of whether the line type causing the creation is <OBJECT> or <OBJECT>:CREATE). The "Field is maintained by Maconomy" column pertains to whether Maconomy automatically assigns a value to the field when creating an entry in the window in question. For instance, Maconomy always assigns order numbers to new orders. The "Standard value applied" result means that Maconomy assigns a standard value to the field. The standard value can be many things, depending on the type of information contained in the field in question. In most cases, the standard value is a system number or information that is transferred from a standard entry, such as a standard item or a standard customer specified in the Company Information or System Information workspace. The standard value can also be a calculated value as in the case of prices on an order line.

Field is a Key Field	Field is Mandatory and Open in Create	Field is Maintained by Maconomy	Result
Yes	Yes	Yes	Not applicable.
Yes	Yes	No	Error, as Maconomy expects an explicit value.
Yes	No	Yes	Standard value applied.
Yes	No	No	Not applicable.
No	Yes	Yes	Standard value applied.
No	Yes	No	If a standard value can be found, it is used. Otherwise, an error occurs.
No	No	No	Blank/zero value applied.
No	No	Yes	Standard value applied.

Using #KEEP while Updating Existing Entries

When updating existing entries, key fields can never contain the #KEEP command, because the entry to be updated must be specified by the key. Entering #KEEP in a key field results in the creation of an entry when using automatic line types.

All other fields with #KEEP, including mandatory fields, retain their current values.

Note that in some cases, mandatory fields require you to enter an explicit value, and you therefore cannot use #KEEP.

Using #KEEP to Use Information from Previous Entry

When you import lines to the sub-tab of a workspace, you must specify (as part of the sub-tab line key) what tab (sales order, item, and so on) the line belongs to. Instead of entering an explicit value on the import line, you can use the #KEEP command to assign the key value from the tab that was most recently imported. This can sometimes be necessary, because in some cases, you cannot know what the key to a given entry should be. For instance, when importing orders and order lines, you may want to structure your import file so that the tab of the order is imported first, and subsequent lines contain information for the order lines of that order. In this case, it is not possible to tell what order number Maconomy will assign to the order, because the order number is retrieved from the number series. Still, each order line must be assigned an order number so that Maconomy knows what order the line pertains to.

When you import key values for new sub-tab lines, the standard value is the value from the entry that was most recently imported to the tab. As in any other case, the standard value is obtained by specifying #KEEP in the field in question. In the preceding example, specifying #KEEP in the **Order No.** field on each order line in the import file therefore means that the order line is assigned the same order number as the nearest preceding order (tab).

You can also use the #KEEP command in the key fields on action lines to perform the action on the tab that was most recently imported.

The following shows an example of this functionality.

Import orders.xls

	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Order No.	Line No.	Item No.	Ordered			
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	#KEEP	#KEEP	1241	4			
6	ORDER LINE	#KEEP	#KEEP	1235	8			
7	SALES ORDERS:APPROVE ORDER	#KEEP						

Sheet1 Sheet2 Sheet3

In this example, the formats of both orders and order lines specify that column B must contain an order number that is part of the key in both types of entry. Line 4 contains information for the tab of the Sales Orders workspace, and the value #KEEP in column B on this line means that Maconomy is to assign the standard value to this field. The standard value for the **Order No.** field in the tab is a number from the number series.

The value #KEEP in column B on lines 5 and 6 also means that the standard value is to be applied to these fields. Because the **Order No.** field is a key field in a sub-tab, the standard value for this field is the value from the corresponding field in the nearest preceding order. Consequently, the line is assigned to the order that was most recently imported.

On line 7, the value #KEEP in the column B specifies that the **Approve Order** action is to be performed on the order that was most recently imported, the order on line 4.

Leaving Fields Blank

If the imported line triggers the creation of an entry, fields that are left blank are assigned the value zero or blank, depending on the field type. As described in “The #KEEP Command,” entering #KEEP causes the field to be assigned the standard value.

If the imported line updates an existing entry, fields that are included in the format but left blank are updated with the value zero or blank. However, the current value can be maintained by entering #KEEP in the imported field.

Fields that are not included in the format are treated as if the field contained the #KEEP value on all lines, both when updating and when creating entries.

The Difference Between Blank, Explicit Values, and #KEEP

This section describes the difference between leaving a field blank, entering an explicit value, and entering #KEEP.

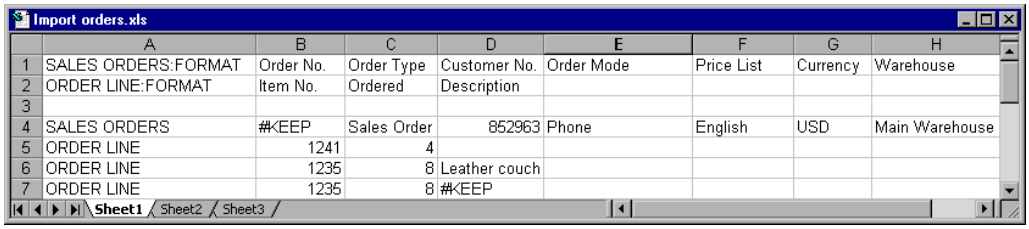
When you leave a field blank, the value that is imported, regardless whether you are updating or creating entries, is always a zero or a blank value, depending on the type of field. This means that if you are updating an existing entry, any value in the field is overwritten with a blank value.

Explicit values, for instance the item description "Leather Couch," are always imported as entered in the import file. When creating entries, this means that the new entry is assigned the specified value. When updating entries, an explicit value overwrites any existing value in that field for the entry in question.

If you enter #KEEP, the value that is imported depends on whether you are updating or creating entries. If the import line in question updates an existing entry, #KEEP means that value in the field in question is not to be changed in Maconomy’s database. If the import line creates an entry, the standard value depends on the type of entry and the setup of the system. Note that in some cases, the standard value is blank or zero.

It is important to note that mandatory fields can never have the value zero or blank in the Maconomy database. Therefore, you must always enter an explicit value or #KEEP in fields that are mandatory, open, and included in the format. Note that in some cases, #KEEP is not valid either, and you must enter an explicit value.

The following example shows an example of an import file for the import of sales orders.


Import orders.xls

	A	B	C	D	E	F	G	H
1	SALES ORDERS:FORMAT	Order No.	Order Type	Customer No.	Order Mode	Price List	Currency	Warehouse
2	ORDER LINE:FORMAT	Item No.	Ordered	Description				
3								
4	SALES ORDERS	#KEEP	Sales Order	852963	Phone	English	USD	Main Warehouse
5	ORDER LINE	1241	4					
6	ORDER LINE	1235	8	Leather couch				
7	ORDER LINE	1235	8	#KEEP				

Sheet1 Sheet2 Sheet3

In this example, assume that all of the lines that are imported will result in the creation of entries. The **Description** field (Column D in the order line format) contains different values on each imported order line. On the first order line (line 5) the item description field contains a blank value. The value on the order line that is created in Maconomy will therefore also be blank, no matter what the normal item description for item 1241 is. The second order line in the import file contains an explicit value, meaning that the order line created in Maconomy as a result of this import line will be "Leather Couch," no matter what the normal description of item 1235 is.

The third order line in this file contains the description #KEEP. This means that the standard value is to be used. The standard value for item descriptions on sales order lines is the description from the item information card, so the order line created in Maconomy will show the normal item description for item 1236.

Leaving Fields Out of the Format

As mentioned in “Line Types,” leaving a field out of the format—that is, not including it on the format line— causes the import program to process that field as if it contained the value #KEEP on each line in the import file. In practice, leaving a field out of the format presents the following advantages:

- When updating existing entries, you only need to include the key fields and the fields that you want to change. The remaining fields keep their existing values. This makes it easier to work with the import file, because the number of columns can be reduced a lot.
- When creating entries, you can leave out all of the fields to which you want Maconomy to assign standard values.
- As described in the “Keys in Table Parts,” the key to most sub-tab lines consists of the key to the tab and a line number that represents the line’s position in the tab. When you import new tab lines, you can therefore usually omit the **Line No.** field from the format, and Maconomy will add each imported line to the bottom of the sub-tab.
- The key to the tab can also be left out of the format for table lines. Maconomy will then assign each line to the tab that was most recently imported, meaning that the line is assigned to the nearest preceding line that represents a tab entry. An example of this functionality is described in “Using #KEEP to Use Information from Previous Entry.”

Reading the Log File

The results of each import attempt are reported in a log file that is generated by the import program. By reading this log file, you can—if there are errors—find out what was wrong with the import file. You can use this information to correct the import file.

You can encounter many different types of errors when trying to import information. However, this section does not list every type of error. Instead, this section focuses on ways to read the log file to weed out the major errors in the file.

After you attempt to import a large file for the first time, the log of errors may be rather extensive. However, when you take a closer look at the actual errors, you will probably see that many of the errors are common to every line in the import file, or to certain line types. By finding out what caused these errors, you can correct the problem, reimport, and get a shorter error log that is easier to read. The following is a list of problems that can cause large numbers of errors in the log file.

Note that in some import programs, you can choose whether logging should be activated. If logging is deactivated, Maconomy stops the import as soon as an error occurs, and the log file thus shows only one error.

The Line Could Not be Imported

When there is an error on a line, the error itself is shown in the log file, but sometimes, the log file also contains an additional line that reports that a line could not be imported. This error message disappears along with the actual error message after all problems on the line have been resolved.

Unknown Line Type

This type of error occurs if you misspell the line type name in the first field on the line. An error occurs for each line on which the line type name was misspelled.

The Entry XX Does Not Exist

This type of error is reported when you refer to an entry that does not exist in Maconomy. An error occurs for each line that refers to the invalid entry.

In some cases, the error occurs if one import line contains errors and subsequent import lines contain references to that entry. Because the first line contains errors, it is not created in Maconomy, and any references to it are invalid. For instance, if you are importing an order followed by a number of order lines, and the order header contains errors, the order is never created, meaning that the references to this order on the subsequent order lines are invalid. After the errors in the tab are corrected, the sub-tab references become valid, and this error no longer occurs.

Practical Usage of Import Programs

This section consists of two parts. The first part describes how to use import programs in certain work procedures. The second part contains a list of hints and tips that can help you save time and trouble when you set up your import files.

Import programs can be used as a tool in a number of different tasks. The most common purposes for using import programs are:

- Changing existing information in your Maconomy system
- Importing new information to Maconomy as a result of
  - Converting paper-based information to entries in Maconomy
  - Changing data from another system to Maconomy
- Performing actions on several entries in the database

This section describes how to create import files to perform these tasks. Note that the procedures that are described are merely suggestions. In your company, you may do things differently, or use import programs for different purposes. The following procedure descriptions should therefore only be considered as guidelines and tips for the considerations that you need to make in different situations.

The Maconomy import programs are advanced tools. Therefore, it is very difficult to anticipate any possible error. As a result of this, an import procedure may take a few attempts where errors occur, and you need to find the cause of the errors.

Changing Existing Information in Your Database

When you must make general changes to existing data, such as payment terms in the customer information, you can in some cases save a lot of time by using import programs.

The change can be made by following these steps:

1. Export the relevant information from Maconomy.
2. Rearrange the information to match the format of the import program.
3. Make the necessary changes in the data.
4. Perform the import.

Export the Relevant Data

When you want to change a number of entries in Maconomy, the first step is to export the information from Maconomy to a tab-separated text file that you can edit and reimport. The export is created by using Maconomy's standard export feature from the relevant Find window.

In the Find window, start by setting up the correct search criteria, making sure that the search results only contain the entries that need to be changed. If you do not set up the right criteria, the search may include a lot of irrelevant entries that you will have to weed out from the exported file. After you have the right search results, use the **Export table** function from the File menu and choose a folder for the exported data. If you use the Maconomy Windows client version 4.0 or

newer, select the columns in the search windows and copy the entries to the clipboard instead, because the **Export table** function is not available in that client version.

Rearrange Information to Match the Import Format

The information that is exported must be changed to match the format that is used by the import program. You must add line types, and perhaps change the field names and change the contents of some fields because the imported values in some cases are not in accordance with the field type rules that are used by import programs.

First, open the exported file in a spreadsheet. If you use the Maconomy Windows client version 4.0 or newer and copied the columns in the search window instead of exporting the table, create a spreadsheet and paste the copied columns into the spreadsheet.

Next, start by inserting a column to the left of column A so that the new column is the first column in the spreadsheet. You need this column to specify the line type of each line. The first line in the exported file contains the names of the fields in the file. This line can be turned into a format line by entering <OBJECT>:FORMAT in the new column that you just inserted. Of course, you need to replace the word <OBJECT> with the correct object name; for instance, if you are working with items, the correct line type name is ITEM:FORMAT. You can find the valid format line names for each program in the manual that is specific to the program in question.

The next step is to change the column headers to match the field names in the manual. This step is not necessary for the programs that allow the use of internal names, because the field names that are exported from Maconomy correspond to the internal names that are used by the import programs. It is very important that the names are entered exactly as in the manual.

The order of the fields is of no importance, but you can change it so that the fields whose values you are changing are at the beginning of each line. That way, it can be easier to see which entries are being changed.

After the column names and order have been changed, you need to make sure that the information in the file is presented in the way in which the import program reads it. For instance, Maconomy exports the values TRUE and FALSE from integer fields if you use the **Export table** action. You need to change this to 1 and 0, using the replace function in the spreadsheet program. Maconomy also exports dates in another format than the one used in the import, and you must therefore change the slashes to dots in all date fields.

Make the Necessary Changes in the Data

Make the appropriate changes in the data—that is, the information that you want to change in the Maconomy database. When making the changes, be sure to observe the rules that apply to different field types. Save the file as a tab-separated text file.

Run the Import

Run the import and check out the results in the log file. The log file is placed in the Maconomy folder unless you specify otherwise in the dialog that appears before the import is performed. If there were errors in the import, read the log file carefully and find out what the problem was. To begin with, the log file can be very extensive, so the easiest way is to look for similar errors on different lines, eliminate one problem at a time, and then run the import again, each time getting a shorter log file. At this point, it is also a good idea to check the Maconomy reference manual for the exact functionality of the fields causing the errors. Keep doing this until the import is successful.

Creating an Import File

If you are starting up on a new Maconomy system with no information, and you need to create a large number of entries in the database, the fastest way to create the entries is to import a file

with all the information. The same procedure can be used any time that you need to create a large number of Maconomy entries based on paper-based transactions, such as time sheets that are submitted by employees or invoices that are received from vendors.

The steps involved in this type of import process are as follows:

1. Examine the functionality of the relevant window.
2. Run a test import.
3. Enter actual data and run the actual import.

Examine the Functionality of the Relevant Window and the Import Program

Before the import, you need to know what default information Maconomy will use when you create entries. For instance, if you are importing customer information, check the standard customer so that you know what information will be transferred to the imported customers when you use the #Keep command or leave fields out of the format. In addition, the dimension values to be derived from imported customers should be decided upon. You should also consider the current setup and system parameters in Maconomy because this may affect the imported fields. For instance, a system parameter can determine whether a value must be manually entered in a field or is maintained by Maconomy, as in the case of the requisition number in the Requisitions workspace. Another example is the import of items, where the setup of the item groups determines whether the items that are imported have inventory control. Running these checks enables you to determine which new entries should have explicit values and which ones can use the default values.

It is a good idea to enter an entry in Maconomy and export it. This allows you to see how the information is structured. However, the columns may need to be moved around, and the headers may need to be changed. In addition, dates and Boolean fields are exported using another format than the one used by the import programs.

In short, you can avoid many problems by knowing the exact functionality of the window into which you are importing data.

Run a Test Import

It is a good idea to run a test import of one or two entries in a new spreadsheet or on the basis of the file that you exported in step 1.

Start by deciding which fields you need to import. Create a format line that contains the names of the fields that you want to import. It is very important that the field names are spelled exactly as in the field list in the program-specific manual. If you can import several line types, for example, information for both the tab and sub-tab of a workspace, enter the format lines in the first lines of the import file. This is not compulsory, but it allows you to use the facilities for freezing panes in Excel. If you are working with frozen panes, you can thus always see the field names, no matter which line you are working on in the spreadsheet.

Enter the relevant information for one or two entries and try to import the file.

Check the log file that is placed in the Maconomy folder unless you specified another location. If the import was successful, start entering the relevant lines and conduct the import.

If there were errors in the import, read the log file carefully and find out what the problem was. When you have resolved the problem, document the problem and the solution, because a similar problem may occur when you try to import the actual data. Try importing the file again after resolving the problem, and see if the import is successful.

Enter Actual Data and Run the Actual Import

After the import of the test data is successful, enter the actual data and try the import. Because some of the information in the actual data will vary from your test import lines, you may encounter some errors. Check with your notes to see if the problems are similar to those in the test data.

To begin with, the log file can be very extensive, so the easiest way is to look for similar errors on different lines, eliminate one problem at a time, and then run the import again, each time getting a shorter log file. Concentrate on the first error in the log file because this error may be the cause of several subsequent errors. At this point, it is also a good idea to check the Maconomy reference manual for the exact functionality of the fields that caused the errors. Keep doing this until the import is successful.

Importing Information Exported from Other Programs

In many cases, you need to import information from other programs. For instance, your carriers may deliver new price lists from time to time, or you are changing from another system to Maconomy. This process is in many ways similar to updating existing entries because you are working with a pregenerated file. However, in addition to the steps that you need to perform when updating existing information, you need to make some extra considerations. A number of these considerations are described in this section.

In some cases, the import of data from other programs takes place on a regular basis, such as once a month. If this is the case, it is a good idea to keep track of all of the changes that you make in the import file. For instance, if a pop-up option has a different name in the output file than in Maconomy, what was it changed from and what was it changed to? This way, you can set up a step-by-step list that describes the changes that are necessary to perform a smooth import, and maybe even set up a macro that makes all the changes for you.

In general, the process of importing information from other programs involves the following steps:

1. Consider the conversion from external fields to Maconomy fields (for example, dimensions).
2. Rearrange the information to match the format of the import program.
3. Run the import.

Consider the Conversion from External fields to Maconomy Fields

The external program probably uses other field names and pop-up values than Maconomy. This, of course, has to be considered. You need to analyze the information and set up algorithms for the conversion. After you have determined how dimension values, fields, and pop-up options should be converted, you can set up a macro that performs the necessary changes. This can be useful if you need to move data on a regular basis, such as once a month. Remember that the setup of, for example, dimension derivation may cause the imported entries to be assigned different dimension values than stated in the import file.

Rearrange the Information to Match Maconomy's Fields

The output format of different field types (date, Boolean, and so on) may not be the same as the input format required by the Maconomy import program. You therefore must make sure that the information in the file is presented in the way that the import program reads it. Check the field formats in the manual against the contents in the import file and make the necessary changes. You may also need to change the field names in the header and move columns around as explained in in the previous description of updating existing entries with import programs.

Run the Import

Run the import and check out the results in the log file. The log file is placed in the Maconomy folder, unless you have specified a different location.

If there were errors in the import, read the log file carefully and find out what the problem was. To begin with, the log file can be very extensive, so the easiest way is to look for similar errors on different lines, eliminate one problem at a time, and then run the import again, each time getting a shorter log file. At this point, it is also a good idea to check the Maconomy reference manual for the exact functionality of the fields that caused the errors. Keep doing this until the import is successful.

Actions on Several Entries in the Database

You may want to perform an action on a number of entries. For instance, you may want to delete entries in your database on a regular basis, for example, to remove any outdated and irrelevant information or as part of a larger work process. Similarly, you may want to approve orders in batches, for instance, once a day.

You can use the Maconomy import programs to perform actions on any number of entries in Maconomy's database. However, the prerequisites that must be met for an action to be performed on a given entry also apply to import programs. For instance, an item can only be deleted if the fields related to quantity in the item's warehouses all have the value zero, if there have been no item changes in the last and the current years, and if no item discount agreements have been created for the current item.

The procedure for performing actions on several entries is similar, regardless of the action in question.

In general, the procedure for performing actions using an import program is as follows:

1. Export the relevant entries.
2. Edit the exported entries.
3. Run the import that performs the action.

Export the Relevant Entries

When you want to perform an action on a number of entries in Maconomy, the first step is to export the information from Maconomy to a tab-separated text file that contains the relevant entries. The export is performed as described in "Exporting the Relevant Data".

Edit the Exported Information

The information that you exported must be changed to match the format that is used by the import program. You need to add line types, change the field names, and change the contents of some fields, because the imported values in some cases are not in accordance with the field type rules that are used by import programs.

First, open the exported file in a spreadsheet. If you use the Maconomy Windows client version 4.0 or newer and copied the columns in the search window instead of exporting the sub-tab, create a spreadsheet and paste the copied columns into the spreadsheet.

Next, insert a column to the left of column A so that the new column is the first column in the spreadsheet. You need this column to specify the line type of each line. The first line in the exported file contains the names of the fields in the file. This line can be turned into a format line by entering <OBJECT>:FORMAT in the new column that you just inserted. Of course, you need to replace the word <OBJECT> with the correct object name. For instance, if you are working with items, the correct line type name would be ITEM:FORMAT. You can find the valid format line names for a given program in the manual for the program in question.

The next step is to remove all columns except for the ones that contain key fields, because only the key is needed to identify each entry on which the action is to be performed. The key fields that are used in the program in question are shown in the program-specific manual. After the irrelevant columns have been deleted, you must change the column headers of the key fields to match the field names in the manual.

It is very important that the names in the file are entered exactly as in the manual.

After the column names have been changed, you must make sure that the information in the file is presented in the way that the import program reads it. For instance, Maconomy exports dates in another format than the one used in the import, and you therefore need to change the slashes to dots in all date fields, if any date fields are part of the key.

Finally, add the <OBJECT>:<ACTION> line type in column A on each line except for the one that already contains the format. Once again, substitute the relevant object name for <OBJECT>, <ACTION>.

Save the file as a tab-separated text file.

Run the Import

Run the import and check out the results in the log file. The log file is placed in the Maconomy folder, unless you have specified another location.

If there were errors in the import, read the log file carefully and find out what the problem was. To begin with, the log file can be very extensive, so the easiest way is to look for similar errors on different lines, eliminate one problem at a time, and then run the import again, each time getting a shorter log file. At this point, it is also a good idea to check the Maconomy reference manual for the exact functionality of the window in question and any special rules about the deletion of entries.

Checklist Before Import

When working with import programs, you can avoid many problems by using the following checklist.

- Check the version. Make sure that the manual and import program match the version and patch number of your current Maconomy system.
- Insert format lines. Make sure that the first line(s) in your import file are format lines that specify the fields to which you want to import information.
- Check the spelling of field names. Make sure that all field names in the format lines are spelled exactly as in the field list in the program-specific manual.
- Check data and field types. Make sure that the contents in each field comply with the values that are allowed as defined by the field type.
- Check the values in mandatory fields. Make sure that all mandatory fields in the format contain a value. This is not always technically necessary, but you can avoid many problems by always entering values in mandatory fields.
- Check for valid references. Make sure that all pop-up values and references to information cards in your import file are valid, meaning that the values to which you refer already exist in Maconomy, and any spelling and punctuation is correct in these references.
- Save as tab-separated file. Make sure to save as a tab-separated text file.

Deltek is the leading global provider of enterprise software and information solutions for professional services firms, government contractors, and government agencies. For decades, we have delivered actionable insight that empowers our customers to unlock their business potential. Over 14,000 organizations and 1.8 million users in approximately 80 countries around the world rely on Deltek to research and identify opportunities, win new business, optimize resource, streamline operations, and deliver more profitable projects. Deltek – Know more. Do more.®

deltek.com