Deltek

Deltek Maconomy®

System Admin Guide

July 1, 2022

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 July 2022.

Contents

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.

How This Document is Organized

This document is organized so that you can go quickly to the section you need. Use Ctrl+F to easily find the term you're searching for, or see the table below to find the section to help you.

Section	Description	Purpose of This Section
Understanding Your System	A detailed description of all the major components of your Maconomy system, and how upgrades and installations work, as well as hardware requirements. We also discuss security considerations for your Maconomy system.	▪ Introduction and high-level details and concepts. ▪ Description of Maconomy, typical setup and components, such as installation server, and so forth. ▪ Helps to acquaint you with terminology and processes used in Maconomy. ▪ Orients new users to the system and its components.
Installation and System Configuration	Provides setup and configurations for your system, including servers, passwords, access control, and fonts, and setup within Maconomy, including User Masking and Scheduled Background Tasks.	▪ Initial setup and configuration. ▪ Provides experienced users with process and steps. ▪ One-time or infrequent use.
System Maintenance and Regular Use	Includes details for performing overall maintenance on your Maconomy system, including database surveillance and regular routines, and steps for regular use, including using data import packages and the MDoc Data Dictionary.	▪ Common procedures and check lists for regular system use. ▪ Details routine system admin duties, such as running regular maintenance, and importing system data.
Extend Your System	Details all the ways you can extend your Maconomy system through different integrations, Maconomy Extender, and localization.	▪ Instructions for integrations. ▪ Procedures that go beyond the usual setup, such as localization steps.
Troubleshooting	Includes troubleshooting as well as frequently asked questions for key areas of Maconomy	▪ Provides a repository for common questions and errors with likely solutions.

Section	Description	Purpose of This Section
	including Maconomy Server and Scheduled Background Tasks.	
COMMANDS	Contains command line section with all Maconomy commands and options.	▪ Provide commands for all tools used outside of development.

Audience

The intended audience for this guide is:

- Maconomy System Administrators
- Database Administrators
- Technical Consultants
- 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.

Revision History / What's New

The table below shows a quick reference of recent changes.

Update	Version	Description
Default Filetypes	2.6	Added new section to list default filetypes in the server.ini that users are allowed to upload.
Running an Index Script	2.6	Adding instructions and examples for running index scripts.
Export to Excel Row Summary	2.6	Added Export to Excel Row Summary.
Native Application Authentication	2.6	Added new section describing the use of OAuth 2.0 Bearer tokens in relation to Azure Native Application authentication.
OAuth 2.0 Bearer Tokens	2.6	Added new section with a generic description of the use cases for OAuth 2.0 Bearer tokens.
Enabling and Disabling SSL	2.6	Removed this section and other outdated instructions.
Web Client debranding	2.6	Removed specific references to iAccess, which is now referred to as Maconomy's web client.
Removed outdated / desupported references	2.6	Removed Deltek Collaboration / Kona section. Removed Portal references.

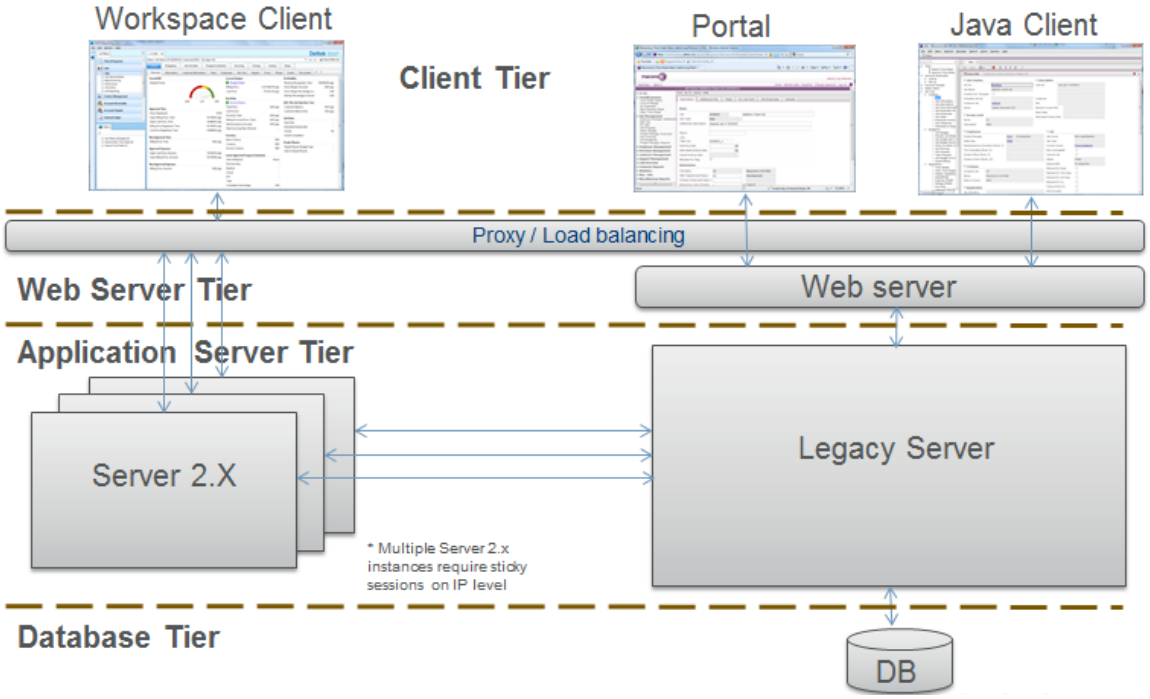
Update	Version	Description
		Removed old Kerberos login configuration, identified as Kerberos SSO.
Analyzer	2.6	Added a section for the new web Analyzer, with link to information on legacy implementation in the Workspace Client.
Increase Allowed Number of Named Users	2.6	Added instructions for customers who need to increase their allowed number of licensed Maconomy users.
Data Import Packages	2.5.4 updated	Updated and added new procedures to the Data Imports Packages section.
Automated Hotfix (CU) Installation	2.5.4	Added new Automated Hotfix (CU) Installation instructions (and removed the outdated instructions).
Printing Right-to-Left Languages	2.5.3	Added Printing Right-to-Left Languages conceptual information and setup instructions.
Okta Support	2.5.3	Added new Single Sign-On with OneLogin support information and procedures.
OneLogin	2.5.3	Updated OneLogin OIDC Metadata URL.
Option Lists	2.5.3	Added description of Option Lists and Selected Values for Data Import Packages.
2FA Enrollment and Reconfiguration	2.5.3	Added Two-Factor (2FA) enrollment and reconfiguration setup steps.
REST API Container Web Access Rules	2.5.2 update	Added REST API Container Web Access Rules, with cross-reference to *Deltek Maconomy Web Services Programmer's Guide*.
Integrate with Web Client	2.5.2	Added instructions to integrate People Planner with the web client.
Removed Background Task information	2.5.2	We have created a standalone Deltek Maconomy Background Task Guide, for ease of use.
File Check	2.5.2	Updated File Check instructions (Simple Magic is no longer supported).
E-Invoicing with Pagero	2.5.2	Added setup instructions for e-invoicing integration with Pagero, as well as some procedures for addressing common errors.
Restructured document and streamlined content	2.5.1	Restructured document for ease of use and logical flow, so that you can read background information if needed, or skip right to setup information or procedures. Additionally, removed outdated material.

Update	Version	Description
		See How This Document is Organized for details.
Scheduled Background Tasks	2.5.1	Added information on Extended Test Beds, Chained Background Tasks, and other updates.
Two-Factor Authentication (2FA)	2.5.1	Added a warning for an iAccess limitation.
Font Administration and Installation	2.5	Updated to add support for multiple fonts, setting default fonts, and a last resort font.
Displaying System Information on Clients	2.5	Renamed the “Menu Dock” section, and added iAccess-specific instructions.
Fonts in MPL2	2.5	Added MPL2 font information.
OneLogin	2.4.5	Added new Single Sign-On with OneLogin support information and procedures.
Data Import Package	2.4.5	Added new Data Import Package information and procedures.
Long Text Support	2.4.4	Added section on the Long Text functionality.
Two-Factor Authentication (2FA)	2.4.4	Updated since iAccess and Touch now support 2FA.
Updating the Maconomy Workspace Client	2.4.4	Added information on disabling the warning message when updating the workspace client.
Scheduled Background Tasks	2.4.3	Added in Email Reporting and Setup, Coupling Service Sliding Panel, Color Coding, and related material.

UNDERSTANDING YOUR SYSTEM

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 new 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.

Standard Maconomy System Naming

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.

Maconomy Cumulative Updates (CU) Naming

The format for Cumulative Update (CU) naming is:

Maconomy <*version*> CU <*nnn*>

For example, the second CU on the 2.2.x stream is now named Maconomy 2.2.x CU 002

Packing Unit (PU) types have a single naming convention:

apu.w2000.sp105a or tpu.NTx86.20_0.p105a?

Legacy PUs are supported without the need for renaming, and MConfig and other tools are able to handle both generations of names.

BPM

BPM name formats include:

RPU Name — rpu.<main version>.<sp>.<cu>.<build number>.<solution>.zip

Example: rpu.20.sp103.cu008.1234.s-std.zip

BIARs — bpm-<product>.<main version>.<sp>.<cu>.<build number>.<solution>.<database>.biar

Documentation of universes — Include version tag. \

ETL analysis — bpm-analysis-etl.<main version>.<sp>.<cu>.<build number>.xml

Example: bpm-analysis-etl.20.sp103.cu0008.1234.xml

ETL peopleplanner — peopleplanner-etl.<main version>.<sp>.<cu><build number>..xml

Example: peopleplanner-etl.20.sp103.cu008.1234.xml

About Maconomy

Maconomy is a project-based enterprise resource planning (ERP) solution that integrates projects, finances and people / resources. It is a financial management solution that provides financial insight with functionality spanning from creating a project to budgeting, evaluating progress to invoicing and project closure. It connect front-office project delivery to back-office financial management, allowing for control over projects. Its reporting and filtering capabilities provide visibility into past performance, current status, and anticipated results. Additionally, it provides global statutory compliance strategy, including a baseline audit, statutory-related functionality, and localization packages to meet country-specific regulatory requirements.

Maconomy has three user interfaces / clients:

- **Workspace Client**—A robust on-premise solution perfect for complex financial functionality, supporting G/L, A/P, and A/R.
- **Web Client**—A browser-based interface with streamlined access to activities such as time, optimized for in-office as well as remote access use.
- **Touch**—A mobile application for easy access to such features as time submission and approvals.

Clients

A Maconomy system can be accessed from different User Interfaces or Clients. Customers are advised which user interface best suits the needs of their Users based on their different user roles. For example, financial and accounting users 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 the web-based client, or a client optimized for mobile, touch-screen devices, such as Touch.

This section provides an overview of the Maconomy clients:

- Workspace Client
- Web Client
- Touch

See Web Client and Touch documentation for more information on these clients.

Workspace Client

The Workspace Client is Maconomy's on-premise client with a customizable system consisting of modules, organized hierarchically around a:

- Business entity (such as a customer)
- Role (such as a project manager)
- Process (such as financial reporting

Workspace Client is comprised of two main areas: Modules / workspaces, and Single Dialogs. The workspaces are where users spend the majority of time. Single Dialogs are intended for consultants / super-users. These dialogs are the components on which (most) workspaces are based, and super-users may use these components to customize their workspaces.

The Workspace Client contains the following features:

- **Workspaces** — Workspaces are a grouping of panels, organized hierarchically around a business entity (such as a customer), a role (such as a project manager), or a process (such as financial reporting). Workspaces are composed of menus, sections, and panels that allow you to interact with specific tools.

- **Search and Filters** — You can use search and filters to find business entities that you need to work with. There are three main search types:
    - Search using filter lists (CTRL+F)
    - Context-specific search (CTRL+G)
    - Manual Search
- **Actions** — Actions are things that you can do. They are similar to functions. An action could be as simple as Print or Delete, or it could be more complicated, such as Print an Invoice.
    - Actions are generally represented as buttons that display in the toolbar at the top of a panel. Each panel has it is own actions, and an action acts on the business entity (or entities) in that panel.
- **Wizards** — Wizards are special windows that help you perform a specific task in Maconomy by guiding you through the steps that are necessary to perform the task.
    - Wizards are used extensively in Maconomy. They are meant to guide you safely through complex tasks, or through common but complicated tasks. When a wizard is used, it is possible to focus on important information, and it is possible to make sure that required field are filled in.
    - An example of a commonly-used wizard in Maconomy is the Create Customer wizard.
- **To-Dos** — It is possible to set up Maconomy so that users are notified when certain events occur. The users are then alerted by a To-Do that displays up in the user's To-Do menu. The To-Dos for each user are grouped into predefined categories. When you double-click a To-Do, the relevant workspace opens.
- **Menu Search Capability** — The Search Capability feature allows you to enter terms into a search field and retrieve information for a particular workspace. This feature lets you navigate the workspace menu easily and search for a specific workspace (in English or its localized equivalent, depending on your language settings). As you type it provides you with a list of suggestions when it cannot find the exact match. You can also carry out a wildcard search when you are unsure about the term you are looking for.
- **Search Favorites** — The Search Favorites feature enables you to take any filter restriction and custom sorting that you frequently use and save it as a ”favorite” for searching purposes. These Search Favorites then function the same way as the predefined searches in the Workspace Client to quickly filter information.

Web Client

Maconomy’s web client is designed for streamlined access to Maconomy functionality supporting front office users such as time and expense activities, projects, invoicing, approvals, etc. it is based on the same business logic as the workspace client but features a more simple, easy-to-use tab design that allows you to click between workspaces. The browser-based application is optimized for in-office as well as remote access use. Look-ups provide quick and easy searches for data and favorites.

Touch

Touch is a mobile app for inputting time, making quick approvals, and recording or authorizing expenses from Android or iOS devices. The app can be downloaded from the Apple App Store and Google Play.

Components / Considerations

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, web client, Touch), as well as the Maconomy Application package.

In addition to these standard components, a customer installation may have a solution installed which is a preconfigured Maconomy installation supporting specific industries.

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

Analyzer

The Analyzer is a powerful and highly customizable reporting tool available in both the web and Workspace clients, although the implementation is different for each client.

The Web Analyzer

Maconomy 2.6 introduced Analyzer reports to the web client. This functionality allows users to make ad hoc queries to the database and view the query results in table form. Specifically:

- The web client now includes workspaces for two reports: a basic and an advanced report template for querying job data.
- Users can utilize these report templates to customize layouts from which they can generate report outputs that fit their specific work requirement.
- At any time, users have the option to view report outputs from reports that they ran previously; they can also export these outputs to an Excel spreadsheet if needed.
- Users also have the ability to export a report layout, allowing others to import this into their application and run it themselves.
- Companies can add new custom reports or existing standard reports to their application.

Access Rights

While the Analyzer report workspaces are available by default, they can only be viewed and utilized by users with the requisite access rights.

Existing steps for granting access rights to workspaces apply. In the Workspace Client, add each report workspace to a user group to grant the users assigned to that group access to the workspace.

You also need to add the report workspace to the web client menu. Use the Extender to modify the menu.json file on the server.

Customization

For information on customization options and related setup, see the *Deltek Maconomy Web Client Install Guide*.

The Workspace Client Analyzer

For information on the Analyzer functionality in the Workspace Client, go to the Analyzer section of this document.

In general, wherever this document refers to the Analyzer, that information applies to legacy implementation in the Workspace Client.

BPM

BPM is another 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 Workspace and web clients.

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.

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.

Servers

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.

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                     |
|  POST /handshake/1         |
|+------------------------->|
|<-------------------------+|
|                           |
|  HTTP                     |
|  POST /workspaces-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.

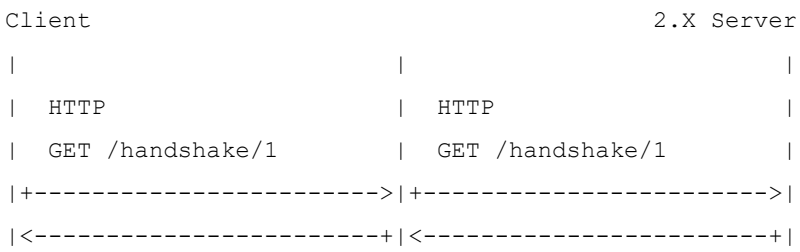
The following are some important parameters that are sent in the handshake:

- The mode of encryption: SSL encryption by intermediary, or 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.

```
Workspace            Reverse Proxy            Maconomy
```



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.

Note: 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

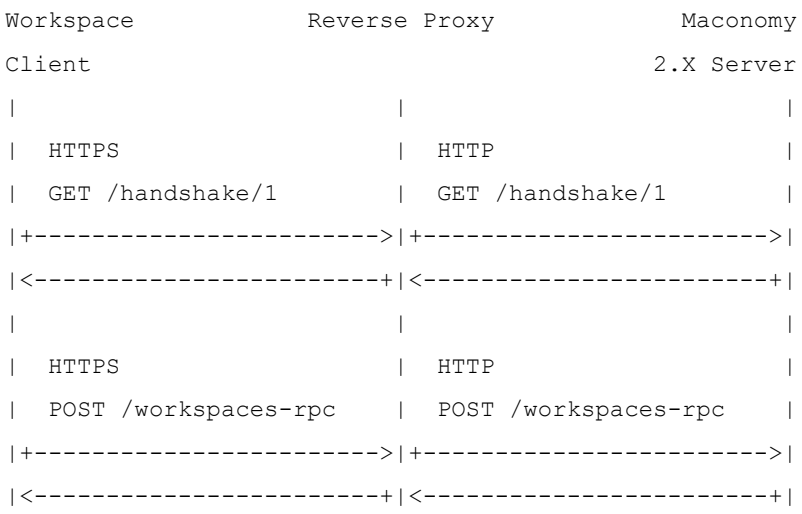
SSL termination is when the proxy server encrypts and decrypts the traffic between the client and the origin server. The proxy server communicates unencrypted with the origin server.

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:

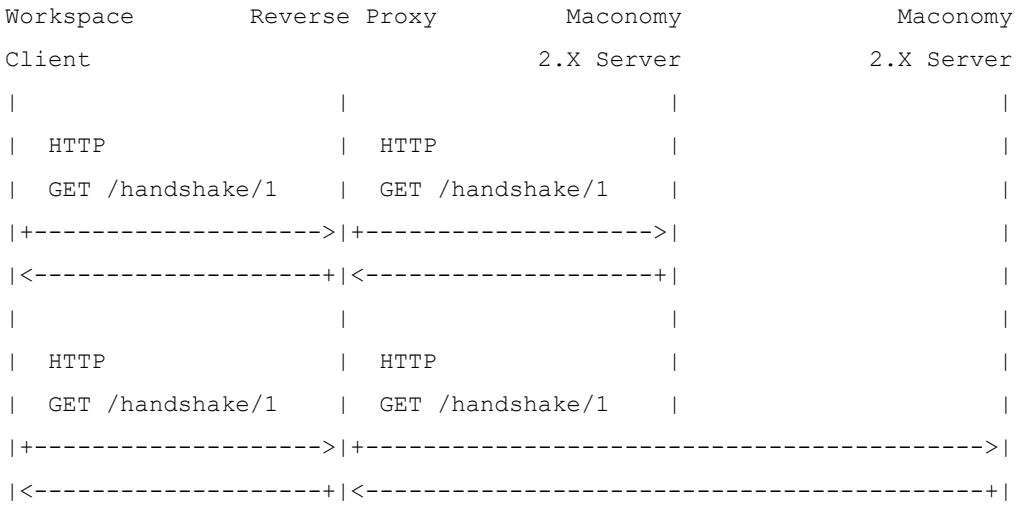
- Increased security because the proxy servers are widely used and actively maintained by companies and/or open-source communities.
- Setup and configuration are well known in customer organizations and fit well with existing infrastructure and network architectures.
- Managing server keys and certificates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

Warning: 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.



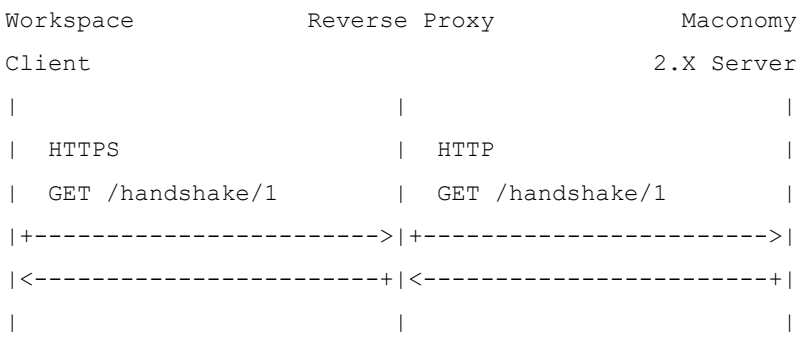
Load Balancing

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.



```
|  HTTPS                    |                          |
|  GET /handshake/1         |                          |
|+------------------------->|                          |
|<-------------------------+|                          |
```

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.

Application Server

This section 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.

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. 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), 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 Upgrade Overview

Default webaccess.ini

A default standard webaccess.ini file is included with all Maconomy installations (including upgrade and new installations). For security purposes, the default file includes restrictions that remove direct access to all search dialogs in Workspace and web clients, but not the search dialogs used by Touch.

The webaccess.ini file is automatically installed by MConfig. It is possible to use Extender to customize the file either adding new restrictions or removing the default restrictions.

Note: Since access to search dialogs is restricted, this could potentially impact integrations.

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.

Hardware Requirements

This document provides information for setting up your Maconomy system and using the Maconomy Hardware Requirements Scaling Tool, which is a dynamic tool that allows you to input specific parameters based on your exact setup, and then generate custom recommendations.

Warning:

- You must engage with a Technical Consultant from Services to use this document and related Hardware Requirements Scaling Tool. *Without this engagement, Deltek takes no responsibility for the resulting output.*
- This document does not include specific reference to hardware offered by vendors or Cloud services (such as Amazon), and it is the responsibility of Services (a Technical Consultant) to advise on this.

Workspace Client Hardware Requirement

Deltek Maconomy offers guidelines for hardware requirements for the client computer running the Workspace Client as minimum requirements and recommended requirements.

The hardware requirements below are valid for the Workspace Client delivered with Maconomy 2.0 and onward. The recommendations may update for later versions of Maconomy.

The hardware specification is detailed by running the NovaBench – System Benchmarking Software version 3.0.4. See www.novabench.com for details.

Minimum Hardware Requirements

The minimum hardware requirements specify a client computer capable of running the Workspace Client without monitoring the response times. This is typically sufficient for those using less complicated functionality, such as time sheets, expense sheets, and mileage reporting.

Following are the minimum hardware requirements for the Maconomy Workspace Client:

- **RAM**: At least 500 MB of free memory (RAM).
- **Configuration**: Depending on the load from other programs running on the machine, a machine configured with 2 GB RAM and 1.6 GHz processor should be the minimal configuration in practice.
- **Hard Disk**: The client program itself plus the files the client program generates usually occupy less than 500MB of disk space. As hard disks typically are much larger, client machine disk space is not expected to become an issue in practice.
- **Screen Size**: The absolute minimum resolution is 1024x768 pixels.
- **NovaBench**: The NovaBench scores follow.
    - NovaBench Overall score: 175
    - NovaBench CPU score: 73

Recommended Hardware Requirements

Using standard Maconomy functionality more complicated than time sheet functionality requires more robust hardware capabilities, and Deltek recommends the following hardware requirements or better.

The following are the recommended hardware requirements for the Maconomy Workspace Client:

- **RAM**: At least 1 GB of free memory (RAM).

- **Configuration**: Depending on the load from other programs running on the machine, a machine configured with 4 GB RAM and 2.5 GHz Intel i5 processor is recommended.
- **Hard Disk**: The client program itself plus the files the client program generates will in most cases occupy less than 500MB of disk space. As hard disks typically are much larger, client machine disk space is not expected to become an issue in practice.
- **Screen Size**: The recommended resolution is 1440x900 pixels or better.
- **NovaBench**: The NovaBench scores follow.
    - NovaBench Overall score: 647
    - NovaBench CPU score: 405

Configuration Introduction

This document covers both vertical and horizontal scalability architecture configurations:

- **Vertical scaling** — Adds more resources (memory, more CPUs, and so on) to a single machine.
- **Horizontal scaling** — Adds more machines.

Choosing between the scaling options is based on individual customer requirements and input from a technical consultant considering many factors, including cost. A single high power server used for vertical scaling may cost 10 times more than 2 servers each with 50% the power, but software licensing could possibly negate this. Also consider that horizontal scaling of the application server using the scalable server of 2.3 GA enables you to have multiple relatively inexpensive servers as opposed to one large one. This configuration also allows for inbuilt redundancy, as well as an inbuilt overhead for failure.

The sample configurations in this document show three single server and three multi-server configurations, and are designed to help in your consideration of future growth.

From 2.3 GA and onwards, you can move from a single server to a multi-server setup at any time, as business requirements grow. However, as you transition from a two-tier to a three-tier installation (which is required for running a scalable server) note that a three-tier installation has an overhead of approximately 20% of the total system usage, which is due to the network connection between the server and database tier compared to utilization of the internal machines' communication when the application server and database are on a single tier. This finding was the result of the performance tests performed on existing Maconomy users using Mercury's LoadRunner® tool. The tests showed that three-tier solutions use TCP/IP-based interfaces (ODBC), which adds an extra communication layer between Maconomy and the underlying database.

Special Actions Required:

- **Discuss with Customer** — Services (a Technical Consultant) should discuss considerations with the customer regarding a horizontal or vertical solution, as there are many parameters to consider which are not covered in this document, such as license and hardware costs, and possible performance issues in specific customer environment, and so on.
- **Approved by PDM —** All use of the scalable server must be approved by PDM prior to implementation, as it is in its early stages of adoption.

Depending on the budget, specific infrastructure bottlenecks and other factors, one solution may be more appealing over another. The response time requirements will decide which solution should be implemented. In general, as hardware becomes cheaper, there may be some financial advantages in choosing a mid-size server for the initial phase and then expanding it once or several times in the first year, instead of paying a higher price outright to make sure that response times are satisfactory for the first one or two years.

One parameter that is necessary for good performance is the disk subsystem. All configurations discussed in this document are configured with high-end internal disk systems, which ensure the fastest data access and optimum stability. For high-end systems with more than 2,000 users, Deltek recommends an external disk system for the database files.

Performance Notes:

- Tiered SANs are often difficult performance-wise for ERP systems because data is moved to slower disks. Maconomy must reside on the fastest tier at all time.
- The number and speed of the disk drives have an impact on the performance of the database.

All sample configurations that are described in this document are designed to handle a standard Maconomy solution installation. These configurations assume that 5% of the total number of users are heavy users, such as back office Finance users, 15% are moderate users, such as Project and Resource Managers, and 80% are light users, such as Time and Expense users, who may access Maconomy using a browser or mobile device. Variance from this split of Users should accordingly be taken into account when sizing.

In addition, the configurations assume a shared-server program pool of not more than four programs per kernel.

Maconomy works with most Intel-based servers.

Note: This document does not discuss backup media, though it is an important part of the technical implementation. In general, the backup system for a Maconomy installation must be an integral part of the company's backup strategy.

Scalability Considerations

Maconomy Infrastructure

The two Maconomy sample configurations below show typical configurations for both a single server and multi-server environment. These configurations can either be installed in a customer's own on-premise setup with physical or virtualized servers, or via hosted virtual environments, such as in Amazon Web Services (AWS) infrastructure.

> **Note:** If you have a configuration utilizing virtual environments, please ensure that all resources like CPU and Memory are dedicated and not shared between instances.

From Maconomy 2.3 and later releases, in order to increase network security, we recommend running the Web server/reverse proxy on the Maconomy Application Server or Servers. This allows running HTTPS across the entire Maconomy stack, covering communication from all Maconomy clients (including the RESTapi) to the Maconomy Server itself. We recommend running the Reverse Proxy in all single or multi-server configurations.

Single Server Environment

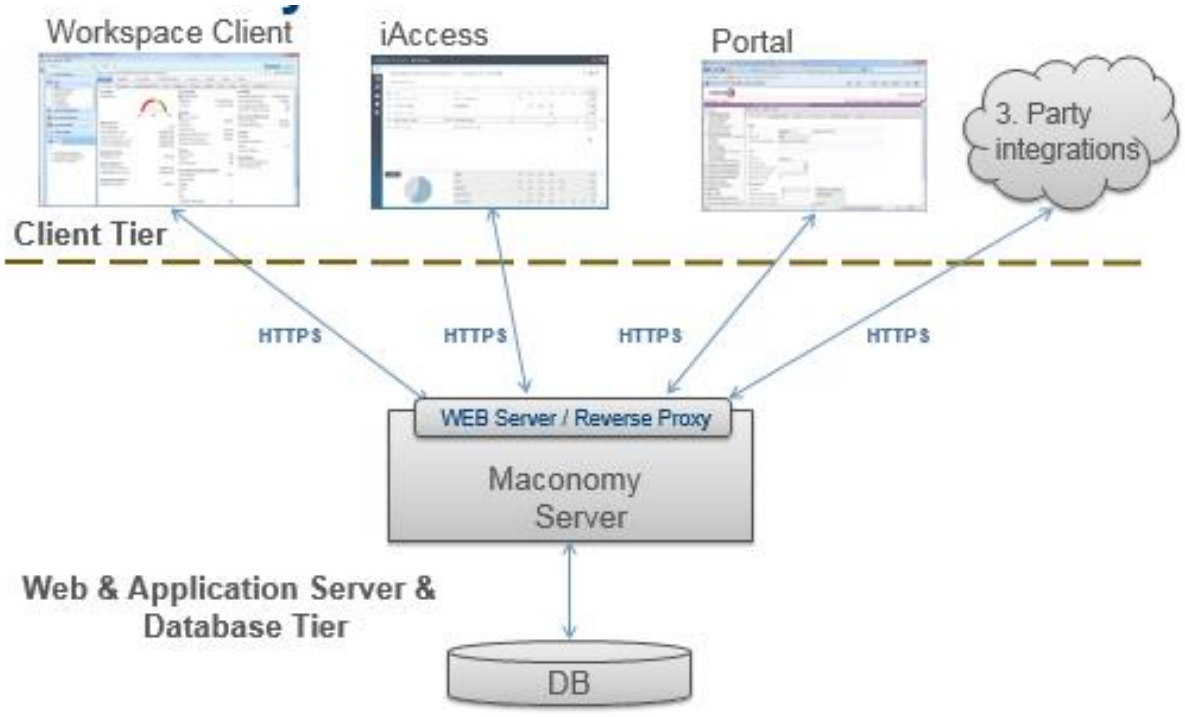
The following example shows a single server Maconomy configuration based on the following parameters:

OS: Linux

Database: Oracle

Clients: WSC, web client, and RESTful API

Users	Memory (GB)	CPU (Kernels)	CPU (Clock)	IOPS	Disk	Configuration *Server_Max*
500	57	27	2.8	375	350	18
2000	61	34	2.8	1500	350	36
5000	70	49	2.8	3750	350	72



Multi-Server Environment

The following example assumes equal configuration and utilization across all servers. Note that it may also be possible and desirable to set up servers without equal load or operations. For example, you may choose to set up one server that is used only for running scheduled tasks. Depending on your needs, you may require a different split of the configuration of resources, with different sizing calculations.

The following example shows a multi-server Maconomy configuration based on the following parameters:

OS: Linux

Application Servers: 3

Peak load per server: 50%

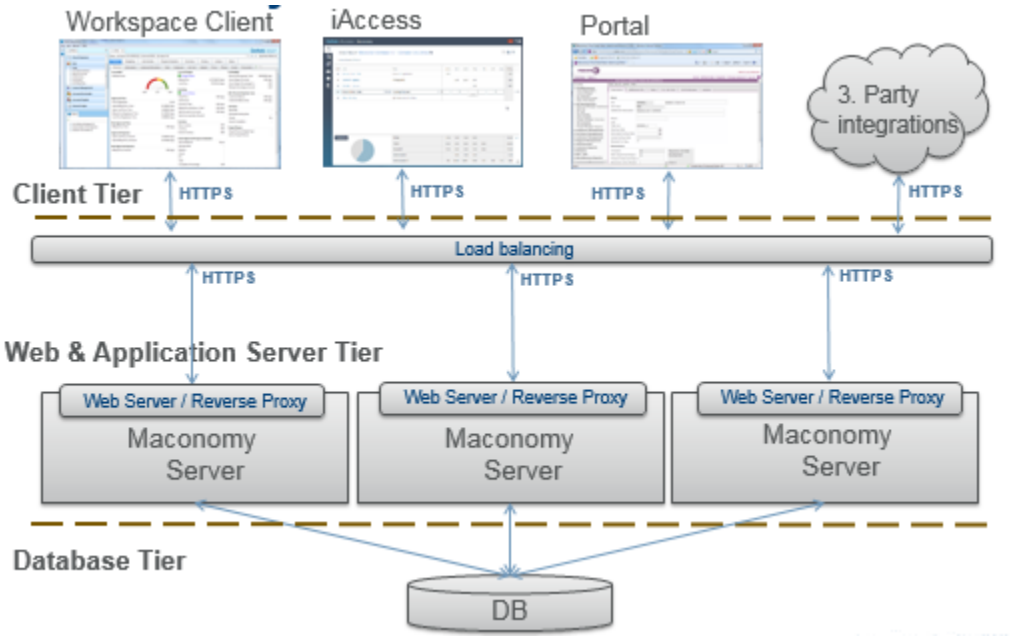
Database: Oracle

Load Balancer: Yes

Clients: WSC, web client, and RESTful API

Users	Servers	Memory (GB)	CPU (Kernels)	CPU (Clock)	IOPS	Disk	Configuration *Server_Max*
2000	Load Balancer	4	2	2.6			
	Application	25	19	2.8		100	
	Database	34	12	2.8	1500	250	24
5000	Load Balancer	4	2	2.6			

Users	Servers	Memory (GB)	CPU (Kernels)	CPU (Clock)	IOPS	Disk	Configuration *Server_Max*
	Application	28	24	2.8		100	
	Database	37	18	2.8	3750	250	
10000	Load Balancer	4	2	2.6			
	Application	33	31	2.8		100	72
	Database	42	28	2.8	7500	250	



Other Scalability Considerations

Databases

Both Oracle and SQL Server have built in capabilities aimed towards horizontal database scalability although both vendors recommend vertically scaling loads of up to 10,000 transactions per second.

As you make server choices, consider that Deltek recommends Oracle (Active) Data Guard and SQL Pier to Pier replication for the Maconomy solution, as they provide only read scalability, and this is relevant for removing the load related to reporting from the transactional database.

Note: Deltek has not certified running specific options, but assumes these options will work with Maconomy through Oracle or SQL Server.

BPM & People Planner Scaling

BPM & People Planner scaling should be considered separately or in conjunction with the recommendation given regarding this document and the Maconomy scaling tool. This could have an impact on Maconomy hardware.

Extra Step with Oracle:

Maconomy is a 32-bit application on the Intel platform. To install Maconomy with an Oracle 64-bit database, you need OracleNet as the interface layer between Maconomy and the database. This adds a data communication overhead of up to 20% to the database. On the other hand, large SQL statements may run faster due to the larger cache size on the database.

This setup is required only for Oracle. SQL Server does not have any special setup requirements.

Virtualization

In general, Maconomy supports systems that run on virtual environments if the OS is certified to run Maconomy. Maconomy does not support the virtual environment itself.

In terms of performance, virtualization translates to additional work for the CPU. The instructions that perform the extra work are normally part of the operating system calls. To support virtualization, an extra layer of OS calls is introduced, and this layer makes up the biggest share of the virtualization overhead.

When running Maconomy on a virtual environment, you must allow room for the expansion of both the CPU and memory. You may also experience a performance degradation (up to 20% in overhead), compared to running Maconomy on a physical machine, due to VM administration, shared I/O, and CPU. The best mitigation is the make sure to run on dedicated resources.

A key challenge for virtualization is the interception and simulation of privileged operations, such as I/O instructions and CPU capacity.

Bandwidth Requirements

This section describes bandwidth testing for Maconomy 2.x+.

This section 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.

This section 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

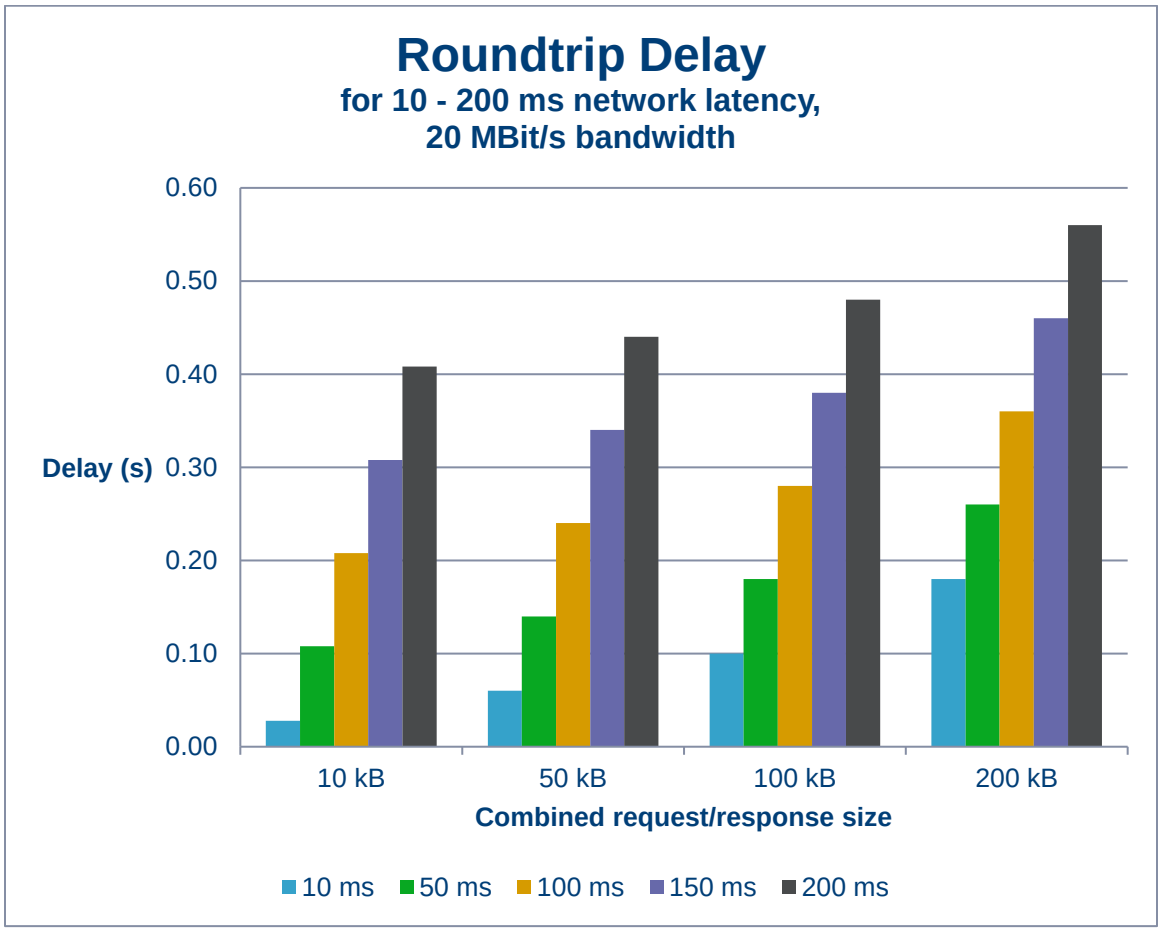
Using a few simple formulas it is possible to calculate the roundtrip delay (RTD) for a request/response pair with a given message size when passing data across a network with a known latency and bandwidth. The RTD is a calculation of the average time it takes to send a request and receive the response,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

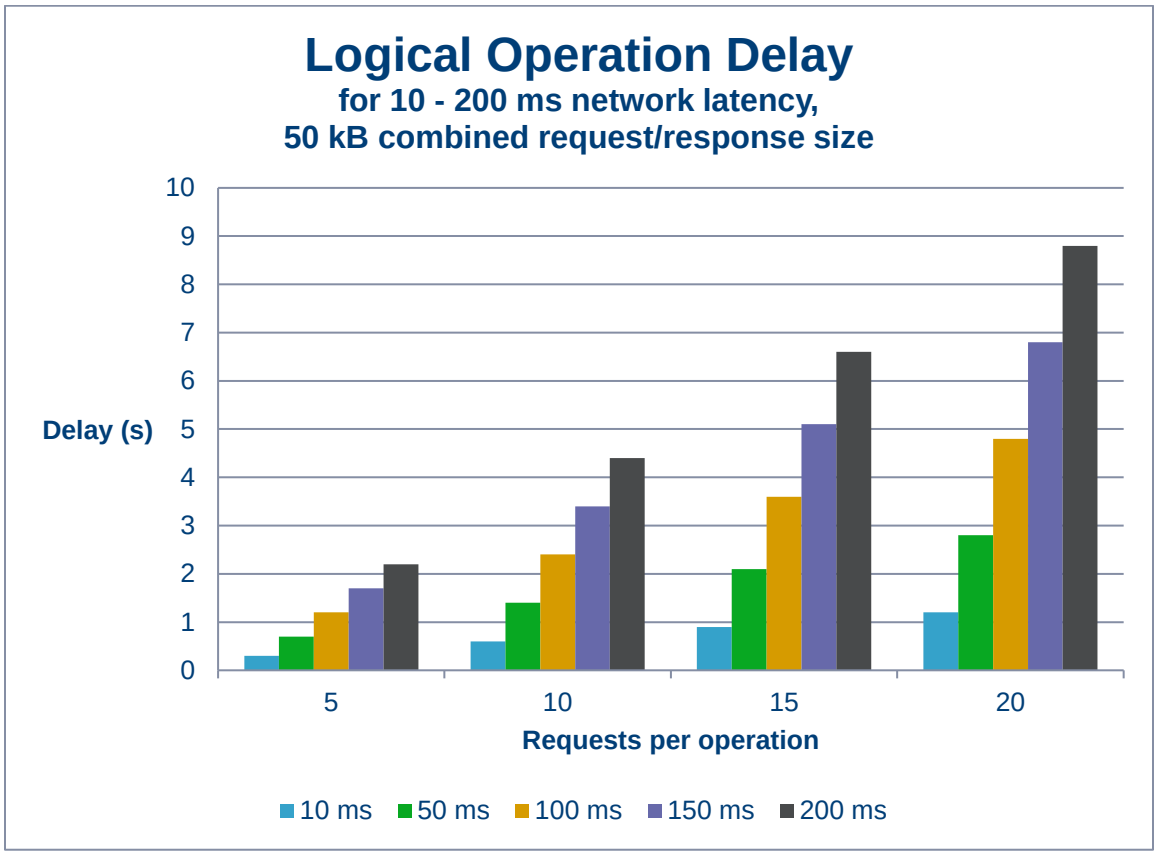
The next figure shows the “logical operation delay” for a complete workflow in the Workspace Client, as a function of the number of request/response roundtrips between the WSC and the CS.

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:

Combined request/response size

Number of users	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.

Security Considerations

This section discusses the following security areas for you to consider:

- Access Control
- Passwords
- SSL

Password Security

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.

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!”

> **Note:** Do **not** use this example, given that it has been published.<Replace text>

If you follow these guidelines, guessing your password should take longer than your password change frequency.

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

Default Filetypes

For security purposes, we limit default filetypes in the server.ini that users are allowed to upload to Maconomy. These types include:

- Composite Document File V2 Document
- HTML document
- Microsoft Excel 2007+
- Microsoft OOXML
- Microsoft PowerPoint 2007+
- Microsoft Word 2007+
- PDF document
- JPEG image data
- PC bitmap
- PNG image
- Zip archive data
- 7-zip archive data
- ASCII text
- ISO-8859 text
- Rich Text Format data
- UTF-8 Unicode
- Unicode text
- XML 1.0 document
- news or mail

Other entries in the server.ini file to support Maconomy functionality are:

- data—Necessary to allow files from support documentation functionality to be saved
- empty—Necessary to allow for files without any content but this is not an allowed filetype

You can modify this list in server.ini if needed.

To modify the default filetypes list:

1. Go to server.ini
2. Find the line:

```
security.filecheck.external.1.allowlist = security-filecheck-
external-allowlist-default.txt
```

3. Either update this file or change to another file (which can be a copy of the default file).

   The file is by default located in the same folder as server.ini but a full path can also be used.

4. Update the relevant file with new lines for file types to be accepted.
5. A new file type can be found by trying to upload a file which is not accepted.

   2 log lines are added in the maconomy.log file in CouplingService\log\coupling folder:

   ```
   WARN  c.m.c.s.s.McFileCheckUsingExternalCommand - File check output
   for file '<filename>': <filetype>, <extra text>
   ```

   ```
   WARN  c.m.c.s.security.McSecurityService - File '<filename>' not
   accepted by: McFileCheckUsingExternalCommand:
   ```

   Add the text <filetype> to the allowlist file.

6. Touch the file server.ini so the allowlist file is read again by the Security Service (or restart the Coupling Service).

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) )
```

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.

REST API Container Web Access Rules

You can specify web access rules for each of the four primary types of container endpoints: specification, filter, data, and field. Addtionally, you can use search as a foreign-key endpoint for each container, dynamically translated to search container requests.

> **Note:** See the *Deltek Maconomy Web Service’s Programmer’s Guide*, “Web Access Configuration” section, for more details.

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, 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.

See the *Deltek Maconomy Web Client Install Guide* for further information.

INSTALLATION and CONFIGURATION

Determine Hardware Requirements

The Maconomy Hardware Requirements Scaling Tool is a dynamic tool that allows you to input specific parameters based on your exact setup. Use this tool to generate custom recommendations.

In order to get the maximum benefit from this tool you must first understand the fundamental differences in configuration (described below). Additionally, resize on a regular basis or as your setup evolves.

Technical Consultants input criteria such as:

- Number of users
- Make up of user profiles
- Number of languages

The tool then output the recommended hardware required in as much detail as possible, including the all key server parameters such as:

- Memory
- CPU (Kernels)
- CPU (clock speed)
- I/Ops

The information the tool provides is completely dynamic, matching the customer's specific requirements.

Before You Begin

Before you begin, you need to complete the following tasks:

- Gather system information
- Consider warnings

Warnings

Virtualization

There is limited guidance around virtualized environments. Plan to take 20% degradation into account.

Network Bandwidth

If you run a three-tier configuration you must have a high-bandwidth, low-latency connection between server and database.

Coupling Service

The Coupling Service is a permanent part of the Application Server and must run on the same instance as the Application Server, not on separate hardware.

Multi-Thread CPUs

Maconomy does not benefit from multi-thread CPUs. Servers with this feature will not benefit from any gains as a result.

Tiered SANS

Tiered SANs are supported by Maconomy architecture, but may be problematic in terms of performance related to the incorrect priority of Maconomy within the SAN and in difficulty troubleshooting.

Unicode Impact

Expect a 15-20% degradation in system performance for all customers moving though the 2.1 release due to Unicode impact. This impacts ALL customers not only those making use of language support for characters now included after conversion.

Sizing Tool Procedures

Server Sizing Guide

Use the steps in this section to use the Hardware Requirements Scaling Tool to calculate specific output for your needs.

Input

To use the Hardware Requirements Sizing Tool, open it with the password *MaconomyRocket!23* In the spreadsheet, complete the following fields in the top Input area.

1. **Users** — Enter the total number of users, such as 20,000.
    a. **Time and Expense Users** — Enter the percentage of the above total users that use time and expense, such as 80%.
    b. **Project and Resource Managers** — Enter the percentage of the above total users that are project and resource managers, such as 15%.
    c. **Finance Users** — Enter the percentage of the above total users that are finance users, such as 5%.

    > **Note on Users:**
    >
    > We split up users in this manner because various users have different load requirements on the system. For example, Time and Expense users exert little load on the system, whereas PMs nominally use more resources working on more complex tasks, and Finance users utilize the system much more aggressively often working with large sums of data (entries) and complex procedures.
    >
    > The number of relevant users displays in gray shading in Column C for each user group.

2. **Languages** — Enter the number of languages used by your Maconomy system. The number of languages affect the amount of memory and CPU usage, as more memory and CPU is required to handle and process each language.
3. **Concurrent users (Max)** — Enter the maximum number of users that normally use the Maconomy system at any one time. Take care to include time zone considerations as well. For example, if half of your organization is working in Asia and the other half is in the U.S., the number of concurrent users at any time may only be 50% of your total number of users.
4. **Environment** — Select whether the environment is physical or virtual.
5. **Server OS (MS / Linux)** — Select whether the server operating systems is Windows or Linux. Note that Microsoft (MS) requires more memory and CPU resources compared to Linux.
6. **Servers (load balanced)** — Enter the number of load-balanced servers on the system.
7. **Peak load per server** — Enter the percentage of the load each server is intended to handle.
8. **DB (Oracle / SQL)** — Select whether you use Oracle or SQL for the database. Both consume similar resources.

9. **Load Balancer** — Select yes or no to indicate whether or not you use a load balancer to dynamically balance the server loads.
10. **Web Client** — Select yes or no to indicate whether or not you use the web client.
11. **RESTapi** — Select yes or no to indicate whether or not you use RESTapi for integrations.

As you input information into these columns, the system recommendations automatically update in the bottom half of the spreadsheet. Ensure that all relevant information is entered.

Recommendations

Maconomy 2.3.x

As you update information in the Input area, the Recommendations automatically display best practice guidelines for these areas:

- Load Balance / Reverse Proxy (WSC)
- Web servers / Reverse Proxy
- Maconomy Server
- Server Configuration
- DB Server (one only)

The recommendations include specifics for the following:

- # (Number)
- Memory GB (each)
- CPU Kernels (each)
- CPU (Clock)
- IOPS
- Disk (GB)
- Configuration (Server Max)

Set Up Server

Setting up an Intermediary Proxy

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

- HTTP traffic
- Workspaces RPC traffic over HTTP/WebSockets

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

The Workspaces 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. In earlier Maconomy versions it was possible to set this up using a “raw” TCP listen port, but the only supported option now is HTTP/WebSockets via the web service endpoint ‘/workspaces-rpc’.

HTTP and HTTP/WebSockets traffic only requires a standard HTTP 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

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.

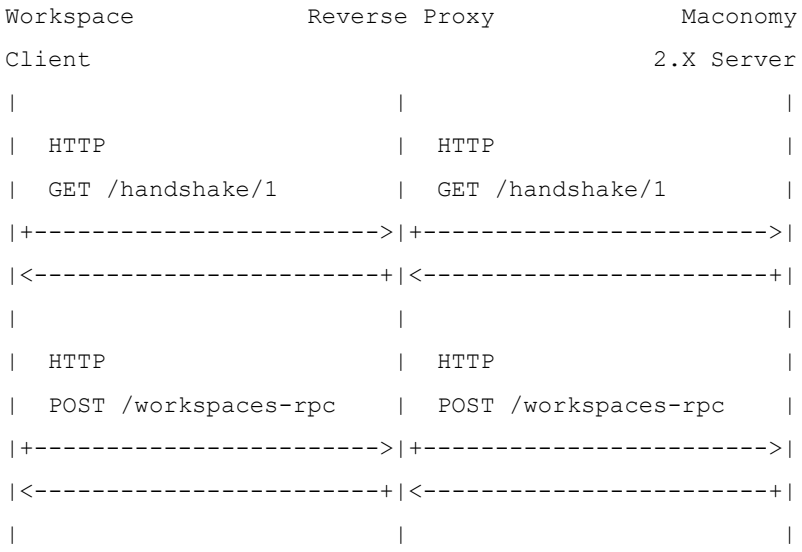
In this example the reverse proxy server machine has the IP address 172.16.1.115, and the Maconomy server machine has the IP address 172.16.1.150.

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.

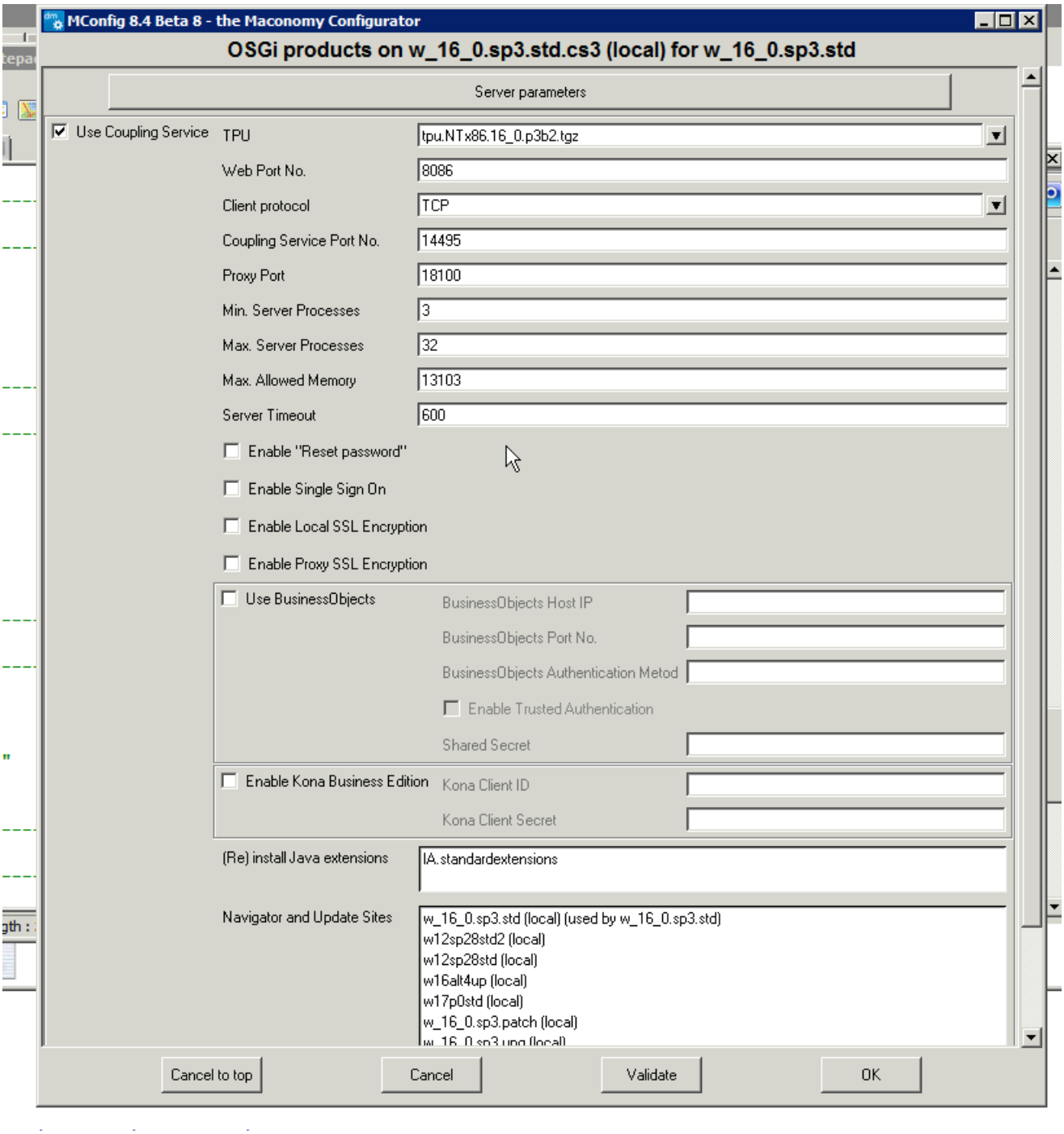


This example uses tcptunnel (homepage) (GitHub).

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
```

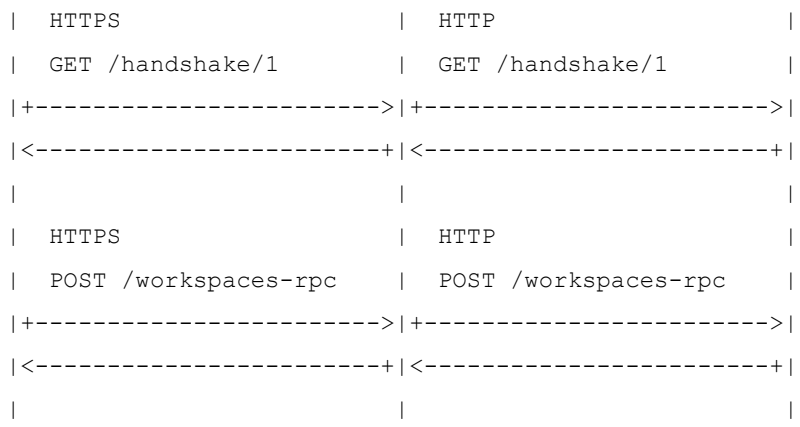
In MConfig, set up the coupling service to use TCP with either no encryption or local encryption:



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
|                          |                        |
```



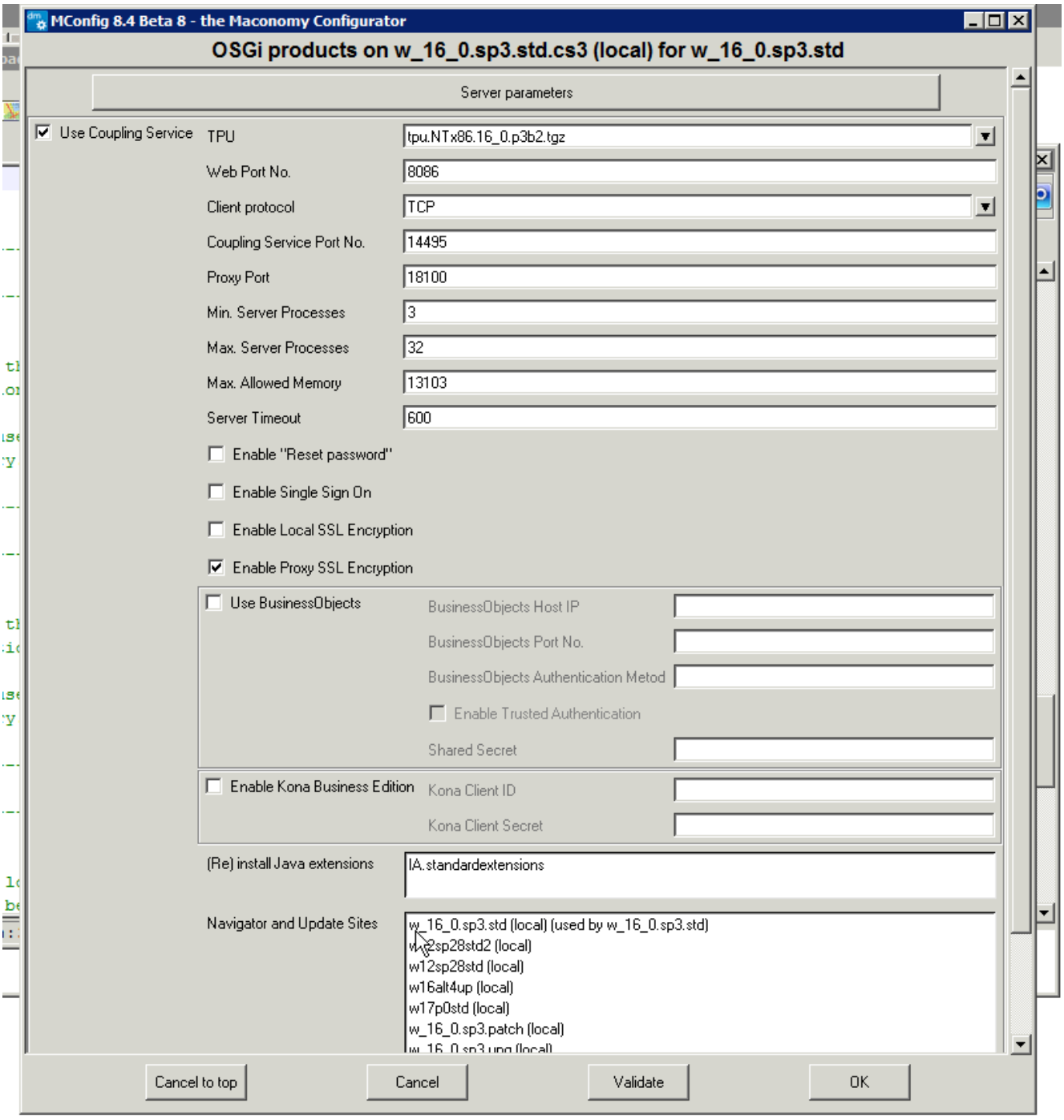
This example uses stud (GitHub)

Run the program with the following parameters:

```
stud --ssl -b 172.16.1.150,18100 -f 172.16.1.115,14495 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.muxrmi.proxy.encryption. There is an optional server.ini parameter called coupling.muxrmi.proxy.port to be used in situations where the proxy server is exposed on a port other than the default port expecting SSL connections.

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

```
coupling.muxrmi.proxy.port = 14444
coupling.muxrmi.proxy.encryption = true
```

This configuration specifies that the proxy SSL (through coupling.muxrmi.proxy.encryption) should be used.

HTTPS Certificate Checking

Note: If the workspace client or Java Analyzer reports an error that explains that it is "certificate related", contact your system administrator for more information.

Issues that are certificate related could be anything related to certificates being out of date, certificates not suitable for the host name used, certificates that cannot be validated by the client or operating system, certificates revoked, and so on.

Most certificate errors are not harmful and are caused by simple client or server setup issues, or administrative issues. However, it is possible that an error indicates a real security issue. Therefore, the best practice is to report any certificate related issues to your system administrator, regardless of what you suspect the cause to be.

Web Proxy Configuration

See the Web-proxy / server configuration documentation for your particular Web-proxy / server product.

Regardless of your Web-proxy product, you want to configure the following items:

- The certificate private & public key certificate files, for example, the .crt and .key files. This is done in different ways depending on the particular Web-proxy / server product used.
- The SSL / TLS version(s) - The WSC only supports TLS version 1.2 in Maconomy 2.4.
- The SSL / TLS cipher suite(s) - The WSC supports the following cipher suites in Maconomy 2.4:
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

  TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- The SSL / TLS compression option(s) - As a general rule, disable TLS compression and a host of other settings specific to the Web-proxy / server product used.

  Note: Ensure that you have procedures in place to renew the certificate(s) used, otherwise access to the system will be denied when the certificate validity period expires. Additionally, make sure that you have procedures in place to revoke the certificate(s) in use if the (web)-proxy server is breached and the certificate private key is stolen.<Replace text>
-

Example

Apache:

This is a basic but fully functional example of how to set up Apache as a HTTPS proxy for the CS.

```
Listen 4000

<VirtualHost *:4000>

ServerAdmin admin@myserver.com
```

```
ServerName myserver:4000

SSLEngine on
SSLProxyEngine on

SSLCertificateFile C:\webserver\certificate.crt
SSLCertificateKeyFile C:\webserver\certificate.key

##############################################################################
# TLSv1.2 setup
##############################################################################

SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
SSLCipherSuite ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256:!aNULL:!MD5
SSLCipherSuite ECDHE-RSA-AES128-SHA256:!aNULL:!MD5

SSLHonorCipherOrder on
SSLCompression      off
SSLSessionTickets   off
##############################################################################
# Proxy Setup
##############################################################################

ProxyPreserveHost On
ProxyRequests off

ProxyPass /workspaces-rpc "ws://localhost:5000/workspaces-rpc" retry=0 disablereuse=On
ProxyPass "/" "http://localhost:5000/" retry=0

<Proxy *:5000>
Order deny,allow
Allow from all
</Proxy>
</VirtualHost>
```

IIS:

<TODO>

- **Host name / IP address checking** - The workspace client in Maconomy 2.4 now checks the hostname and / or IP-address of the server address / hostname used against the certificate as part of the HTTPS certificate checks. This means that a certificate appropriate for the server address / hostname must be used.
- **Validity checking** – The workspace client in Maconomy 2.4 now checks the validity of the server certificate as part of the HTTPS certificate checks. This means that a valid server certificate must be used at all times.
- **OCSP / CRL checking** – The workspace client in Maconomy 2.4 now checks whether the server certificate has been revoked as part of the HTTPS certificate checks. This check is only done if the certificate contains OCSP and / or CRL information. You can disable this check by editing the Maconomy.ini file of the workspace client and removing the following options:

  -Dcom.sun.security.enableCRLDP=true

  -Dcom.sun.net.ssl.checkRevocation=true

  The changed workspace client must then be distributed to the client machines. Disabling the OCSP / CRL checks is useful if the network performance impact of the OCSP / CRL checks is too high.
- **Distributing self-signed certificates to client machines** - The workspace client uses the list of certificates from the personal OS certificate stores on Mac OS X, Mac OS, or Windows to determine trust. This implies that if the certificate used by a Maconomy system is a self-signed certificate which cannot be validated using the existing (root) certificates in the Mac OS X, Mac OS, or Windows certificate stores, then that certificate or its root issuer needs to be added to those.

Install Scalable Server

Scalable Server enables you to set up multiple Maconomy application server nodes accessing the same database to increase performance and scalability.

A Scalable Server setup consists of:

- One Master Node
- A number of Replica Nodes

Performing a Scalable Server installation involves installing Maconomy on the master node and replicating the setup to the replica nodes. These steps are all performed in MConfig.

Before performing the setup using MConfig, there are a number of prerequisites that must be checked. Furthermore, each node in the setup must be prepared for the setup. These tasks are described in the following sections.

Prerequisites

A Maconomy Scalable Server installation requires all nodes to be more or less identical. This is necessary because the master node's configuration is replicated across all replica nodes. This includes, for example:

- Server 2.x (Coupling Service) configuration for number of server processes to launch.
- Server 2.x memory settings.
- File system paths to database client software.
- Installed Maconomy extensions.

It is therefore necessary to observe the following prerequisites:

- All nodes must run the same operating system (OS). A mixed OS setup is not supported.
- All nodes must have the same amount of memory (RAM) and disk space.

After preparing the nodes for installation (see next section) the following should be observed:

- The database client software is installed in the same paths on all nodes.
- The user maconomy (Linux) or Maconomy (Windows) must have the same home directory on all nodes.

Preparing the Nodes

Each node (the master node as well as every replica) must be prepared as described in the following sub sections.

The steps are:

1. Create the Maconomy OS user. (not necessary on master node)
2. Install database client software.
3. Install OpenSSH and rsync.
4. Create SSH key pairs on the master node.
5. Configure SSH for public key authentication on the replica nodes.

Note: A simple way to prepare all nodes is to prepare the master node (inluding creating the Maconomy OS user1) and then clone the master node into the desired number of replica nodes. This is particularly easy in a virtualized environment.

Creating the Maconomy OS User

Each node must have a Maconomy OS user. This user is used for running all Maconomy processes including installed services.

Note: On the master node the Maconomy OS user can be created automatically by MConfig. However, in order to use the master node as template to clone when settings up replica nodes, you must create the Maconomy OS user manually.

MConfig automatically creates the Maconomy OS user on the master node but you must manually create the user on all slave nodes.

Linux

To create the Maconomy user on Linux, please follow these steps.

Note: It is assumed that the home directory of the Maconomy OS user is /data/maconomy. If another home directory is wanted please change the value of the variable MACONOMY_HOME_DIR in the commands below.

Login as root and at the shell prompt type the following commands to create the Maconomy OS user:

Note: The # sign and the preceeding characters is the shell prompt shown when logged in as root and it should therefore not be typed

```
[root@slave ~] # export MACONOMY_HOME_DIR=/data/maconomy

[root@slave ~] # useradd -d $MACONOMY_HOME_DIR -m -s /bin/csh maconomy [root@slave ~]# passwd
maconomy

Changing password for user maconomy. New password: ********

Retype new password: ******

passwd: all authentication tokens updated successfully.
```

Next, we create and setup the directory /usr/maconomy which is used to hold Maconomy configuration.

```
[root@slave ~] # mkdir /usr/maconomy

[root@slave ~] # chown maconomy:maconomy /usr/maconomy [root@slave ~] # chmod 700 /usr/maconomy

[root@slave ~] # touch /usr/maconomy/initmaconomy

[root@slave ~] # chown maconomy:maconomy /usr/maconomy/initmaconomy [root@slave ~] # grep
1^mac:1 /etc/inittab > /dev/null || \

echo "mac:2345:wait:/usr/maconomy/initmaconomy > /dev/null" >> /etc/inittab
```

Next, we setup the Maconomy OS user's custom login script.

```
[root@slave ~] # cat << EOT > /usr/maconomy/maconomy.login alias ll "ls -l"
alias h history set history=200 set filec=1
set prompt="\$LOGNAME \! % " umask 022
setenv PATH $MACONOMY_HOME_DIR/bin:\$PATH EOT
```

Please make sure to enter all the lines beneath the command as well. They are not output from the command but input to the command.

Finally, we *source* the custom login script from the default login script.

> **Note:** When sourcing a script from another script it is executed as if it was inlined in the other script.

```
[root@slave ~] # echo "source /usr/maconomy/maconomy.login" >> ~maconomy/.login
```

This completes setting up the Maconomy OS user on Linux.

Windows

To create the Maconomy user on Windows, please follow these steps:

1. Create a local user named “Maconomy”.
2. Set a secure password for the user as this user will have administrator privileges.
3. Set the user’s password to never expire.
4. Add the user to the local group “Administrators”.
5. Grant the user the privilege “Log on as a service”: Run “Local Security Policy”, navigate to “Security Settings” > “Local Policies” > “User Rights Assignment”, “Log on as a service” and then add the user “Maconomy”.
6. Create the directory to be used as Maconomy Server Home – typically “C:\maconomy”. This must match the directory used on the master node.
7. Change the permissions on the directory granting the user “Maconomy” full access.
8. Change the ownership of the directory to make it owned by the user “Maconomy”.

Install Database Client Software

See the *Deltek Maconomy Install Guide* for details.

Install OpenSSH and rsync

A Maconomy Scalable Server setup uses Secure Shell (SSH) and rsync for replicating the initial setup and future changes from the master node to the replica nodes.

SSH provides a secure encrypted channel for between nodes which allows MConfig on the master node to execute commands on the replica nodes. Furthermore, the rsync command provides efficient replication of files across the established SSH connection.

On Linux, SSH and rsync are components that are installed with the operating system (OS). During OS install, please select to have an SSH server installed (named “SSH Server” or “sshd”).

On Windows, SSH and rsync are components that are not provided by Microsoft. Deltek recommends using Cygwin to get these components installed. Please remember to update the OpenSSH package whenever an important update is released.

Installing Cygwin on Windows

1. Start the installation by downloading the Cygwin installer from this location:

   https://cygwin.com/setup-x86_64.exe

2. Launch the installer. You can follow these steps or divert from them if your node or Internet connection requires you to do so.

> **Note:** This could be the case if you need to install to a different location and/or drive or if your Internet connection needs to go through a proxy.

3. Select “Install from Internet”.
4. Choose “C:\cygwin” as Root Directory for Cygwin.
5. Select “Install for All Users”.
6. Choose “C:\Users\Maconomy\Downloads” as Local Package Directory.
7. Select “Direct Connection” as Internet connection.
8. Choose a download site – preferably one close to your physical location. An HTTP-based mirror is preferred over FTP.

   You will now arrive at the package selection screen.

To select OpenSSH for installation:

1. Type “openssh” in the search field and locate the package “openssh: The OpenSSH server and client programs” under the “Net” category.
2. Select the “openssh” package for installation by clicking the text “Skip” next to the package. The text will change to the version scheduled for installation.

To select rsync for installation:

1. Type “rsync” in the search field and locate the package “rsync: Fast remote file transfer program” under the “Net” category.
2. Select the “rsync” package for installation by clicking the text “Skip” next to the package. The text will change to the version scheduled for installation.

To complete the Cygwin installation:

1. Click **Next** at the package selection screen.
2. Click **Next** at the “Resolving Dependencies” screen.
3. The installer will now download and install the selected packages.
4. Click “Finish” to complete the installation of Cygwin.

Configuring the Cygwin OpenSSH server (sshd) on Windows

After installing OpenSSH as part of Cygwin the OpenSSH server needs to be configured on all replica nodes.

> **Note:** For details about setting up OpenSSH under Cygwin please refer to the file /usr/share/doc/Cygwin/openssh.README which can be located from the Cygwin bash prompt.

If the Maconomy Extender is going to be used for deployments – which always go to the master node – OpenSSH server must also be configured on the master node.

Please follow these steps:

1. Open a Command Prompt that runs as Administrator – i.e., with elevated privileges.
2. Run the command C:\\cygwin\\bin\\bash

You are now at the bash prompt. To install the OpenSSH server (sshd) run the command ssh-host-config and answer the questions as described below. Please note that most of the output has been cut out below. The cut out parts have been replaced by the text <snip>. For a complete transcript of an ssh-host-config session, please see Appendix X.

```
$ ssh-host-config


*** Info: Generating missing SSH host keys
<snip>
*** Query: Should StrictModes be used? (yes/no) yes
<snip>
*** Query: Should privilege separation be used? (yes/no) yes
<snip>
*** Query: new local account 1sshd1? (yes/no) yes
<snip>
*** Query: Do you want to install sshd as a service?
*** Query: (Say "no" if it is already installed as a service) (yes/no) yes
*** Query: Enter the value of CYGWIN for the daemon: []
<snip>
*** Info: No privileged account could be found.


*** Info: This script plans to use 1cyg_server1.
*** Info: 1cyg_server1 will only be used by registered services.


*** Query: Do you want to use a different name? (yes/no) no
*** Query: Create new privileged user account 1NODE\\cyg_server1 (Cygwin name: 1cyg_server1)?
(yes/no)
*** Info: Please enter a password for new user cyg_server.    Please be sure
*** Info: that this password matches the password rules given on your system.
*** Info: Entering no password will exit the configuration.
*** Query: Please enter the password:
***  Query: Reenter:
<snip>
*** Info: Host configuration finished. Have fun!
```

To allow SSH connections to the node the Windows Firewall setting must be adjusted:

- Launch “Windows Firewall with Advanced Security”.
- Select “Inbound Rules” in the left pane.
- Click “New Rule. . . ” under “Actions” in the right pane.
- Select “Port” as rule type and click “Next”.
- Select “TCP” and enter “22” for “Specific local ports” and click “Next”.

- Select “Allow the connection” and click “Next”.
- Select that the rule should apply only to “Domain” and “Private” networks and click “Next”.

> **Note:** This is the suggested setting but it should be reviewed by the customer’s information security department and could depend on the concrete network topology.

- Enter “sshd port 22” for the name and “Firewall rule to allow inbound SSH connections.” for the description of the new firewall rule. Click “Finish” to complete the new firewall rule.

We can now start the OpenSSH Server service by issuing the command net start sshd at the Command Prompt (running as Administrator):

```
C:\\>net start sshd

The CYGWIN sshd service is starting.

    The CYGWIN sshd service was started successfully.
```

Please verify that you can log into the node from another machine using password authentication. A successful login session could look like this:

```
[maconomy@master ~] $ ssh Maconomy@replica
Maconomy@node1s password:
Maconomy@replica ~

$ exit # logout again
[maconomy@other-node ~] $
```

This completes setting up the OpenSSH Server on Windows. We are now ready to configure SSH public key authentication which is necessary regardless of which OS is used.

Create SSH Key Pairs on the Master Node

In order for MConfig to be able to log into the replica nodes from the master node with SSH using public key authentication we need to create a key pairs consisting of a private and a public key on the master node. The private keys are kept on the master node and are protected by passphrases8. The public keys are distributed to all replica nodes.

MConfig needs both to connect as the Maconomy and the Administrator/root OS users on the replica nodes. To accomplish that, we generate two SSH key pairs on the master node: one for the Maconomy OS user and one for the root (Linux) / Administrator (Windows) OS user.

> **Note:** Passphrase is just another word for a long password.

Creating the SSH key pair for the Maconomy OS user

To create an SSH keypair use the command ‘ssh-keygen’ from a bash shell logged in as the Maconomy OS user. Please note the parameters forcing the generation of a RSA key pair with a key length of 2048 bits. You can choose a longer key length if you desire.

Use the default location for the key pair which is in the .ssh folder in the home directory of the Maconomy OS user.

Please select a strong passphrase as this is protecting the private key.

```
[maconomy@master ~] $ ssh-keygen -t rsa -b2048 Generating public/private rsa key pair.

Enter file in which to save the key (/home/maconomy/.ssh/id_rsa): Enter passphrase
(empty for no passphrase):

Enter same passphrase again:
```

```
Your identification has been saved in /home/maconomy/.ssh/id_rsa. Your public key has
been saved in /home/maconomy/.ssh/id_rsa.pub. The key fingerprint is:
SHA256:buc55UDFnrTpsRyx0t7Qrhi+8ekmD0A5mC5B3ipjaDI maconomy@master The key1s randomart
image is:
+---[RSA 2048]----+
|    .     .    |
|   o . o .  =   |
|   o + +  = B   |
| .  + . .o @ .  |
|E.+ o . S. = B  |
|oo o . . .o * o |
|       o.+* .   |
|      . o==+.   |
|         +B=    |
+----[SHA256]-----+
```

The key pair is now created and ready for use.

Creating the SSH key pair for the root OS user (Linux only)

If we are installing a system on Linux, please repeat the procedure from the previous section this time logged in as the root user on the master node.

Configure SSH public key authentication – replica nodes

Configuring a replica node for public key authentication involves these steps:

- Copy the SSH public key generated on the master to the replica.
- Verify public key authentication.
- Disable password authentication on replica.

Copying the SSH public key to the replica node – Maconomy user

To copy the public key from the master node's Maconomy account to the replica node's Maconomy account we use the ssh-copy-id command. You need to enter the password of the Maconomy OS user on the replica to copy the SSH key.

```
[maconomy@master ~] $ ssh-copy-id maconomy@replica

/usr/local/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
any that are

/usr/local/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to i

maconomy@replica1s password: Number of key(s) added:      1


Now try logging into the machine, with:    "ssh  1maconomy@replica1" and check to
make sure that only the key(s) you wanted were added.
```

To verify that the key was copied properly to the replica node we will now attempt to login and this should succeed without being asked for a password. You might, however, be asked for the passphrase of the private key.

```
[maconomy@master ~] $ ssh maconomy@replica

Last login: Fri Dec 23 10:40:32 2016 from 192.168.1.2


[maconomy@replica ~] $ exit # logout again Connection to maconomy@replica
closed. [maconomy@master ~] $
```

Copying the SSH public key to the replica node – root user (Linux only)

If we are installing a system on Linux, we now need to repeat these step for the root user. Start by logging in as the root user on the master node.

To copy the public key from the master node's root account to the replica node's root account we use the `ssh-copy-id` command. You need to enter the password of the root OS user on the replica to copy the SSH key.

```
[root@master ~] # ssh-copy-id root@replica

/usr/local/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
any that are

/usr/local/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to i  root@replica1s  password:


Number of key(s) added:        1


Now try logging into the machine, with:    "ssh
1root@replica1"  and check to make sure that only
the key(s) you wanted were added.
```

Please note that this command might fail if root logins are not allowed on the replica which is often the case per default. In that case you will need to copy the public key manually to the root user's ~/.ssh/authorized_keys file and observe that this file must have the permissions 0600.

To verify that the key was copied properly to the replica node we will now attempt to login and this should succeed without being asked for a password. You might, however, be asked for the passphrase of the private key.

```
[root@master ~] # ssh root@replica

Last login: Fri Dec 23 10:42:42 2016 from 192.168.1.2


[root@replica ~] # exit Connection to root@replica closed. [root@master ~] #
```

Disabling SSH Password Authentication

We now disable password authentication on the replica to ensure only public key authentication is allowed. Please edit this file.

- /etc/ssh/sshd_config(Linux)
- /etc/sshd_config (Windows/Cygwin)

Find the commented out line #PasswordAuthentication yes, duplicate it and modify the copy so the file contains these lines:

```
#PasswordAuthentication yes PasswordAuthentication  no
```

Now, reload the SSH Server configuration. On Linux run this command as root:

```
[root @master ~] # service sshd reload
Reloading sshd:     [      OK     ]
```

On Windows run this command in bash started from a Command Prompt running as Administrator (elevated privileges):

```
Maconomy@replica  /cygdrive/c
$ net stop sshd && net start sshd The CYGWIN sshd service is stopping.
The CYGWIN sshd service was stopped successfully.


The CYGWIN sshd service is starting.
The CYGWIN sshd service was started successfully.
```

Finally, we want to verify that password authentication is disabled. We do that by trying to log into the replica from the master using forced password authentication. The login attempt must fail like this:

```
[maconomy@master ~] $ ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no
maconomy@repl Permission  denied (publickey,keyboard-interactive).
```

This concludes configuring SSH public key authentication.

Disabling root login on slave nodes after installation (Linux only)

If you for security reasons do not want to allow root login to the slave nodes after the installation has been completed, you should edit this file:

- /etc/ssh/sshd_config(Linux)

  Find an existing line containing PermitRootLogin and modify it or if not found add it so it reads:

  ```
  PermitRootLogin no
  ```

Now, reload the SSH Server configuration by running this command as root:

```
[root @master ~] # service sshd reload
       Reloading sshd:     [      OK     ]
```

Installing the Master Node

The master node is installed using MConfig precisely as you would create a Maconomy Standard or PSO installation.

When the installation is completed continue to the next section.

Set Up Nodes

Use the Maconomy Server Node window to define new nodes for an application. Maconomy Server Nodes lists servers which will be synchronized for this application. Nodes can be added and deleted from the list.

When adding a node, MConfig checks that you can establish an SSH connection to the node server as a user called "Maconomy" ("maconomy" on Unix).

There are two ways to work with nodes:

- *Add new nodes to an existing application* — Use the **Synchronize node** button to bring the node up to date and start Maconomy Daemon and Coupling Service.
- *Change the configuration of an application* — All nodes are synchronized when changes are installed on the "main" Maconomy server.

You can only add nodes with the same operating system as the "main" Maconomy server.

Initial Setup

You must create and install an application before it can be synchronized to other servers. After you create the application, then you can set up nodes.

When first installing Maconomy, follow these steps to set up nodes:

1. Configure the first server as usual, including the database, as described in Standard installation or PSO Installation in the *Deltek Maconomy Install Guide*.

   > **Note:** You can set up nodes when creating an application, before the actual installation. Then, nodes are synchronized when the application is installed.

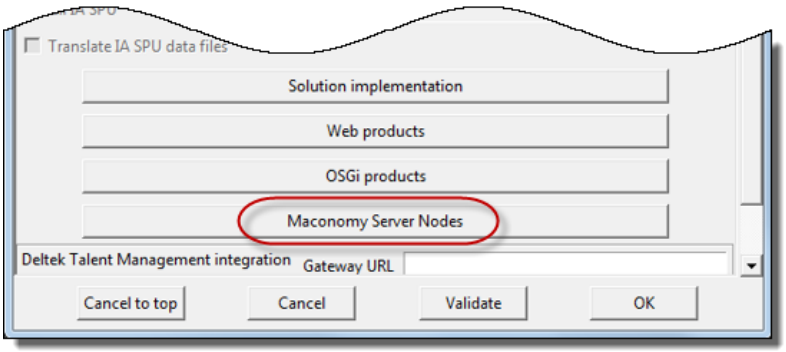
2. Add New Nodes as described below.
3. Click **Synchronize node**.

Adding New Nodes

Use the steps below if you need to set up several new servers or install other configuration changes as well.

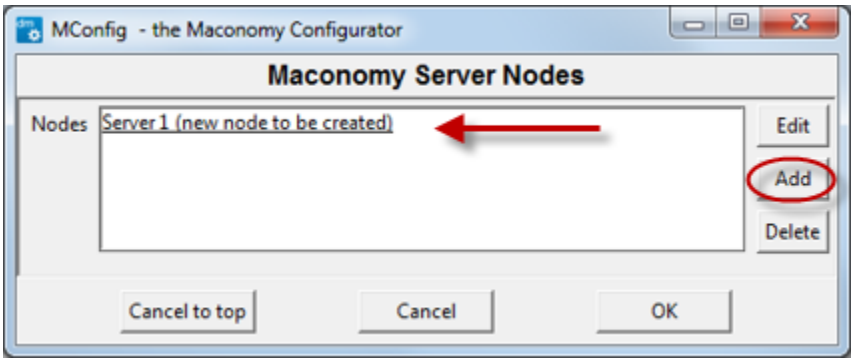
To add new nodes:

1. On the Application Instance window, click **Maconomy Server Nodes**.



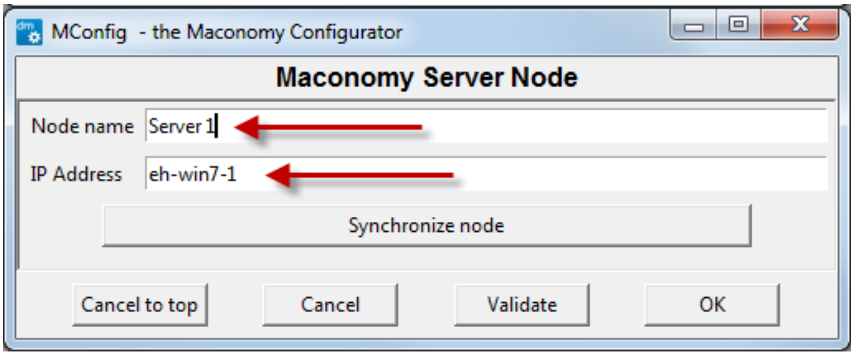
   The Maconomy Server Nodes dialog displays.

2. Click **Add** to set up a new node.



The Maconomy Server Node dialog displays.

3. Enter a name and the IP address of the new node and click **OK**.



4. Add more nodes as needed, and click **OK** when done.

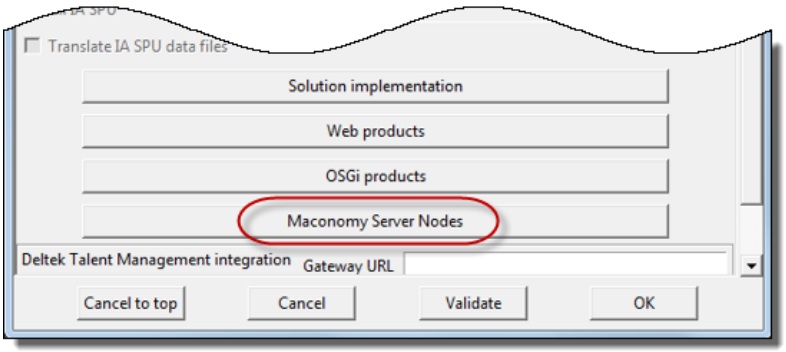
Changing Configuration

Use the steps below if you quickly need to set up a new server for an application.

> **Warning:** Once you have created the nodes, MConfig will synchronize the Maconomy setup. Do not make manual changes in the configuration files on the node servers, as these will be lost in the synchronization.

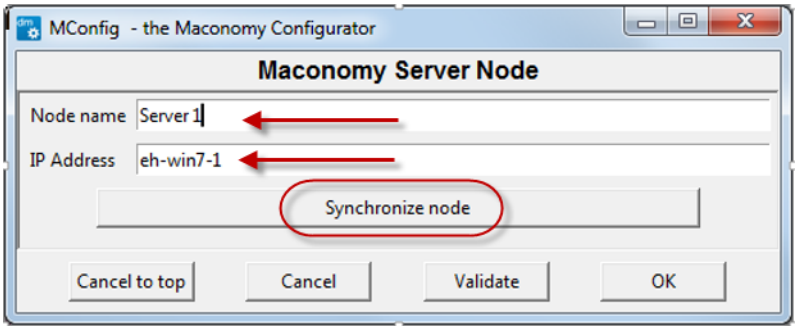
To change the configuration of nodes:

1. On the Application Instance window, click **Maconomy Server Nodes**.



The Maconomy Server Nodes dialog displays.

2. In the Nodes description area, click **Delete** to if a node must be deleted and/or click **Add** if a new node must be created.
3. If you added a node in step 2, you next add the node name and IP address, then click **Synchronize node**.



4. Click **OK**.

Silent Sign-In

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.

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\
    ```

    > **Note:** 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.

Single Sign-On

The Single Sign On (SSO) add-on to Maconomy allows users to use an external authentication provider when logging into Maconomy. The list of supported identity providers include Microsoft Active Directory and Microsoft Azure, as well as other OpenID Connect (OIDC)-capable authentication providers such as OneLogin and Okta. With SSO enabled, the Maconomy Clients will either log in automatically, or show a vendor specific login screen when started.

SSO can be set up in two ways. With name matching, a check is performed to see whether the user's network username matches the login name of an existing user in Maconomy.; if the login name exists, the user is logged in automatically. With name mapping, the user's network username and domain are checked against the configured network usernames and domain entries in the Maconomy database. If a user with a matching network user name and domain exists in Maconomy, the user is logged in 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 Maconomy login screen, where he or she must enter his or her Maconomy username and password.

Configuration

The Maconomy server must also be configured to use SSO. This is done in MConfig, on the OSGi product configuration screen for the Maconomy application you wish to configure. In the 'Domain login method' dropdown you select the login method relevant for your scenario:

- **Kerberos account login:** SSO against Microsoft Active Directory where you directly specify the account credentials for the Service Principal Name (SPN).
- **Kerberos keytab file:** SSO against Microsoft Active Directory using a Java keytab file to store the account credentials for the SPN.
- **Azure OpenID:** SSO using an OIDC application defined in Microsoft Azure.
- **Generic OIDC:** SSO using a different OIDC authentication provider.

After this you will be presented with the method-specific settings that must be configured for each login method.

Add-On Number

To use SSO, the installation must be updated with an installation number and add-on set number 79.

Single Sign-On with Kerberos

The Single Sign-On (SSO) add-on to Maconomy allows users to use their local domain credentials to log into Maconomy. If the user is running the Maconomy client on a domain-joined computer the login can normally be performed automatically, otherwise the user will be prompted to enter the network username and password that is associated with the user's Maconomy account.

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]

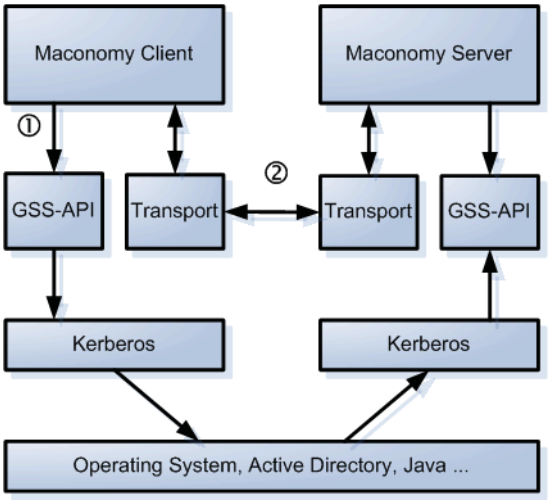
Kerberos was developed by the Massachusetts Institute of Technology (MIT). Maconomy uses the Java GSS-API (Generic Security Services API) to decode and consume Kerberos tickets produced by any compatible Kerberos-based authentication provider. This includes Microsoft Active Directory for the PC and Mac platforms supported by Maconomy. The Workspace Client on Microsoft Windows also supports the native Windows SSPI library for a more seamless SSO experience.

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, 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).

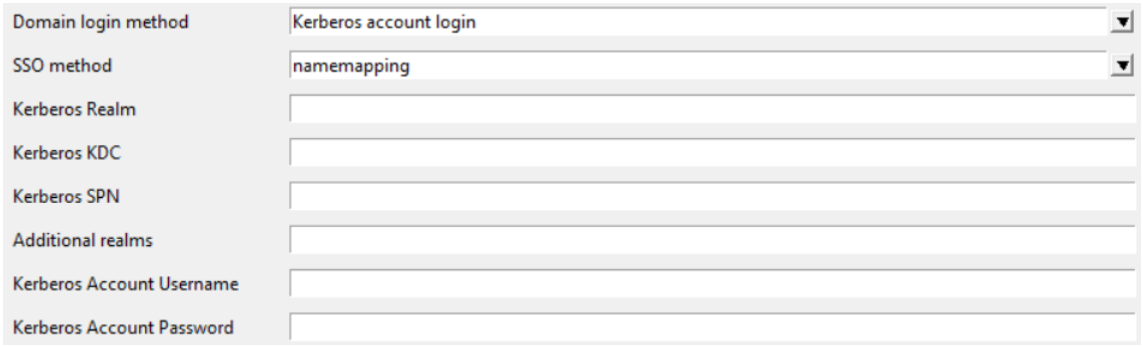
Setup Instructions

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 Maconomy to authenticate users, it must be logged in to this account. The login information is configured in the SSO configuration in MConfig, either directly or through a Java keystore file.

[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

1. 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 also configured in the SSO configuration in MConfig:In the **Domain login method** field, select **Kerberos account login** or **Kerberos keytab file**.

   If you select **Kerberos account login**, the following fields display:

   If you select **Kerberos keytab file**, instead of the two Kerberos Account fields you will see:

2. SSO method: namemapping or namematch, see description in the Single Sign-On section above.
3. Kerberos Realm:This is the name of the realm (domain) under which the Maconomy service has been registered, for example:

   ```
   example.com
   ```

4. Kerberos SPN: This is the name of the service Maconomy that has been registered as the authenticator, for example:

   ```
   Maconomy/test.example.com
   ```

5. Kerberos KDC: This is the network name or IP address of the Kerberos KDC (Key Distribution Center).
6. Additional realms: 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.
7. Kerberos Account Username: This is the account name of the service account associated with the Kerberos SPN.
8. Kerberos Account Password: This is the password for the service account.
9. Kerberos KeyTab file: This is a Java keytab file containing the encrypted password for the service account. See the documentation for the Java ktab tool for details:

   https://docs.oracle.com/javase/8/docs/technotes/tools/windows/ktab.html

Browser Setup

Maconomy web clients use the SPNEGO protocol to perform Kerberos SSO. The SPNEGO protocol will try to negotiate an authentication context between the server and the browser which satisfies the security requirements of both sides. Usually this works seamlessly, but in some scenarios the browser might be unable, or unwilling, to fulfill the security requirements of the server. In such cases you can try to follow the steps below to manually set up your browser to perform Kerberos-based SPNEGO SSO.

Browser Setup for Internet Explorer, Edge and Chrome

For Internet Explorer (IE), Microsoft Edge and Google Chrome, you may need to add the Maconomy webserver address to the Local intranet zone, as Windows might not permit Kerberos-based SPNEGO SSO for untrusted websites.

More details on using SPNEGO 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

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. Click **OK** to accept the changes.
5. Close the **about:config** page.

Macintosh Setup

The Maconomy Workspace Client 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.

Single Sign-On with Microsoft Azure

Maconomy supports Single Sign-On using Microsoft Azure as the authentication provider. To set up SSO through Azure, you must:

- Configure Azure
- Configure Maconomy

Setup Instructions

This login module is disabled by default. Once enabled, Azure SSO is deployed across all interfaces.

To configure the Maconomy Azure OpenId login module, perform the following procedures:

Sign Up for a Microsoft Azure AD Account

If your firm does not already have a Microsoft Azure AD account, you can sign up for a free account at https://azure.microsoft.com/en-us/free.

Microsoft also offers an Azure AD Premium account for a cost at https://azure.microsoft.com/en-us/trial/get-started-active-directory.

The Premium edition is not required for using the single sign-on solution for Maconomy, unless you need to enable Azure multi-factor authentication and/or conditional access restrictions.

Configure Azure

Create a Maconomy Application to Azure

The process requires the following application to be set up in Azure:

- Maconomy – Azure Web application for Maconomy Server, Maconomy clients (including Workspace and web clients), and Touch native and Web application

In this process, you must add Maconomy applications to Azure Active Directory and obtain Client ID.

Tip: Open Notepad and keep it handy to copy and paste IDs.

Create New Registrations

You must register Maconomy applications within the Azure Tenant ID.

To create a new registration in Azure AD:

1. Log in to the Azure portal at https://portal.azure.com/.
2. Log in to Microsoft Azure and Click New registration.
3. On the Register an application blade, in the Name field, enter a descriptive name such as Maconomy Server.
4. In the Supported account types field, select Accounts in this organizational directory only – [name] Single tenant.
5. In the Redirect URL field, select Web.
6. In the related URL field, enter https://<directory>.onmicrosoft.com/maconomyserver.

   Example: For XaYbZc Engineers, enter https://xaybzc.onmicrosoft.com/maconomyserver.

   Verified URLs have a green check mark beside them.
7. Click **Register**.

Show the Tenant and Client IDs

Show the Directory (tenant) and Application (client) IDs, as this information is needed later in the process. These are the values that must be put into MConfig at a later stage.

To view the tenant and client IDs:

1. On the App registrations > Maconomy blade, take note of the IDs.
2. Copy each the Application (client) ID and the Directory (tenant) ID, and paste in Notepad.

Configure Client Redirect URLs

To add redirect URLs:

1. On the **Authentication** blade, in the **Redirect URLs** area, click **Add URL**. A new URL line displays.
2. Add redirect URLs for Touch and web clients.

> **Note:** Tab after entering each URL. A green check mark will appear to confirm that the format of the URL is valid.

3. Add the following reply URLs for each of the client applications that will use Azure for authentication:

   Workspace Client:

   https://login.microsoftonline.com/common/oauth2/nativeclient

   Web Client:

   https://<webserver-host-and-root-path>/oauth

   Touch Web Application:

   https:/<webserver-host-and-root-path>/maconomyshared/backend/oauth2authorizereturn.php

   Touch Native Application:

   https:/<webserver-host-and-root-path>/maconomyshared/backend/oauth2authcodereturn.php

4. Click **Save**.

Configure Maconomy

To enable the OIDC login module and link OneLogin users to Maconomy users:

1. In MConfig, go to the OSGi Products window.
2. In the **Domain login method** field, select **Azure OpenID**.

   Once you select this option, the following fields display:

   Enter the Tenant ID, Client ID and Client Secret obtained from Azure, as described previously.

Native Application Authentication

Native Application authentication is an Azure offering for non-interactive authentication scenarios. Unlike standard end user authentication, Native Application authentication typical occurs through either application access tokens or certificate-based authentication, and this login process produces an OAuth 2.0 Bearer token rather than an OIDC authorization code credential.

To authenticate Native Application and submit an authentication request to an endpoint:

1. In Azure, go to the OAuth 2.0 ‘oauth2/token’ authorization endpoint.
2. Ensure the `resource` parameter is set to the id of the Maconomy Azure application as specified in the Client ID setting in the Maconomy SSO configuration,
3. Ensure the `client_id` parameter is the identity value of the native application itself.

   Example

   Below we include a template for such a request, encoded as a ‘Curl’ command. Values in curly braces must be replaced with the values specific to your installation:

```
curl -X POST https://login.microsoftonline.com/{tenant-id}/oauth2/token ^
-F Content-Type=application/x-www-form-urlencoded ^
-F host=login.microsoftonline.com ^
-F scope=https://{tenant-url}/.default ^
-F client_id={native-application-id} ^
-F grant_type=client_credentials ^
-F resource={maconomy-client-id}
```

For your native application to authenticate itself against Microsoft Azure, add the relevant authentication specific parameters in order. See Microsoft's documentation on the native app authentication flow for further details.

4. The return value from a successful call to this endpoint is a JSON object containing an `access_token` string property. Submit the value of this property can be to the Maconomy RESTful API in an `Authorization: Bearer {access_token}` request header to complete the login process.

Additional configuration of the Maconomy Azure OpenID login module is required in order for Maconomy to accept Native Application Bearer tokens. Specifically, configure the login module to recognize the application identity claim of a Native Application as a valid user claim.

To configure the login module:

1. Deploy a custom configuration file named `maconomy.security.ini` to the `Custom/Definitions` or `Custom.{shortname}/Definitions` folder of the Maconomy system. This file contains extra parameters for the installed login modules.
2. Override the `userIdentity` parameter for the login module named `MaconomyAzureOIDCLoginModule` to include the `appid` user claim. This isin addition to the default user claims, which are `upn` and `unique_name`:

   ```
   MaconomyAzureOIDCLoginModule {
     userIdentity = upn,unique_name,appid
   }
   ```

   Once this change is deployed, the login module recognizes the `appid` user claim and looks for a Maconomy user with matching identity:

   - If the chosen SSO Method is namemapping, the Maconomy user must have its Network Username field in the User Role Information window set to the application identity value, while the Domain Name field must be blank.
   - If the SSO Method is namematch, the Maconomy user must instead have the application identity value set as its Login Name, with the exact same casing as the value sent by Azure.

Single Sign-On with OneLogin

The third-party vendor OneLogin is supported by Maconomy. It offers a cloud-based identity and access management system that works across all Maconomy user interfaces, for both cloud and on-premises deployments.

To set up OneLogin, you must:

- Configure OneLogin
- Configure Maconomy

Setup Instructions

This login module is disabled by default. Once enabled, OneLogin is deployed across all interfaces.

To configure the Maconomy generic OIDC login module with OneLogin as the authentication provider, perform the following procedures:

Sign Up for a OneLogin Account

If your company does not already have a OneLogin account, go to https://www.onelogin.com/ for information on free trials, package pricing, and support documentation.

Configure OneLogin

Create a OneLogin OIDC Application

To create a OneLogin OIDC application:

1. Log in to your OneLogin account ('<account-name>.onelogin.com').
2. Click **Administration** to access your administration dashboard ('<account-name>.onelogin.com/admin').
3. Click **APPS » ADD APP**.
4. Search for **OIDC**, and select **OpenID Connect (OIDC)**.
5. If this is your first OIDC app, leave all values at default.
6. Click **Save**.
7. Go to the SSO tab.
8. Copy your Client ID and Client Secret into a text document.
9. From the **Token Endpoint** drop-down list, select **POST**.
10. Click **Save**.

Configure a Client Redirect URI

To configure a Client Redirect URI:

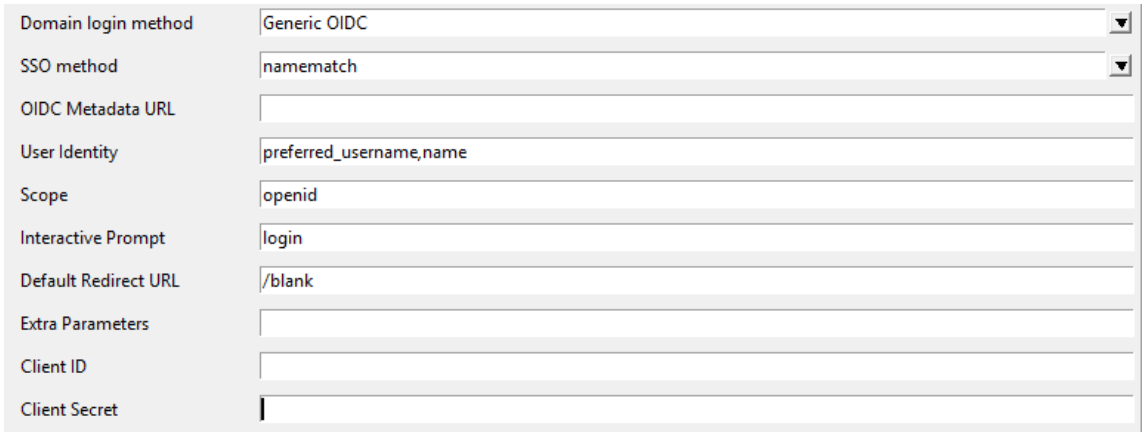
1. Go to the OIDC app page.
2. Click **Configuration**.
3. In the **Redirect URIs** field, enter the URI of the page that will receive the OIDC access token. It can take up to five minutes before you can use the new redirect URI.

> **Note:**
>
> - 'http://' URIs are only permitted for 'localhost' for test purposes. If you need to test OneLogin on a Maconomy installation hosted on a different machine, you have to configure a TLS proxy for it.
> - **Do not** end specified redirect URIs with a '/'.
> - Always add the default redirect URI: 'https://<webserver-host-and-root-path>/blank' or 'http://<webserver-host-and-root-path>/blank'.

Configure Maconomy

To enable the OIDC login module and link OneLogin users to Maconomy users:

1. In MConfig, go to the OSGi Products window.
2. In the **Domain login method** field, select **Generic OIDC**.
    a. Once you select this option, the following fields display:

b. 

c. **Note:** MConfig automatically fills out some of the fields with default values. Edit these if needed.

3. Fill out these fields. Except for **Extra Parameters**, all fields are mandatory.
   a. In the **SSO method** field, choose one of two options: **namematch** or **namemapping**.
   b. In the **OIDC Metadata URL** field, enter the following value: https://{account-name}.onelogin.com/oidc/2/.well-known/openid-configuration.
   c. Skip the **Extra Parameters** field. You do not need to specify any parameters for OneLogin.
   d. In the **Client ID** field, specify your OIDC client ID.
   e. In the **Client Secret** field, specify your OIDC client secret.
   f. Accept the default values provided for the rest of the fields.

g. **Note:** Customers on Enterprise Cloud who want to activate OneLogin will need to send this information to Deltek Cloud Operations via a support case.

4. To save your changes, click **OK**.
5. In the succeeding dialogs that display, click **OK** or **Next** (whichever applies) to install your changes.
6. In the Workspace Client, go to the workspace that exposes the User Role Information container (for example, the Users workspace).
7. Link OneLogin users to Maconomy users. This depends on the value you selected for the **SSO method** field in the OSGi Products window of MConfig.

- If you specified **namematch**, make sure the names of your OneLogin users and the names of your Maconomy users match.
- If you specified **namemapping**, enter the **user principal name (UPN)** or email address components of your OneLogin users ('user@domain') as Network Username and Domain Name in Maconomy.

Single Sign-On with Okta

Okta is an OIDC authentication provider. It offers a cloud-based identity and access management system that works across all Maconomy user interfaces, for both cloud and on-premises deployments.

To set up Okta, you must:

- Configure Okta
- Configure Maconomy

Setup Instructions

This login module is disabled by default. Once enabled, Okta is deployed across all interfaces.

To configure the Maconomy generic OIDC login module with Okta as the authentication provider, perform the following procedures.

Sign Up for a Otka Account

If your company does not already have an Okta account, go to https://www.okta.com for information on free trials, package pricing, and support documentation.

Configure Okta

Create an Okta OIDC Application

To create an Okta OIDC application:

1. Log in to your Okta account ({account-name}.okta.com)
2. Click **Applications** in your developer console (https://{account-name}.okta.com/dev/console)
3. Click **Add Application** and select **Web**. Click **Next**.
4. Enter the host and port of the Maconomy system in the **Base URIs** field.
5. Enter the redirect URIs for each of your client interfaces (WSC, web client, Touch). For the WSC this is: https://{host}:{port}/blank
6. Ensure **Allowed grant type** is set to **Authorization code**.
7. Click **Create**.

Configure a Client Redirect URI

To configure a Client Redirect URI:

1. Go to the OIDC app page.
2. Click **Configuration**.
3. In the **Redirect URIs** field, enter the URI of the page that will receive the OIDC access token. It can take up to five minutes before you can use the new redirect URI.

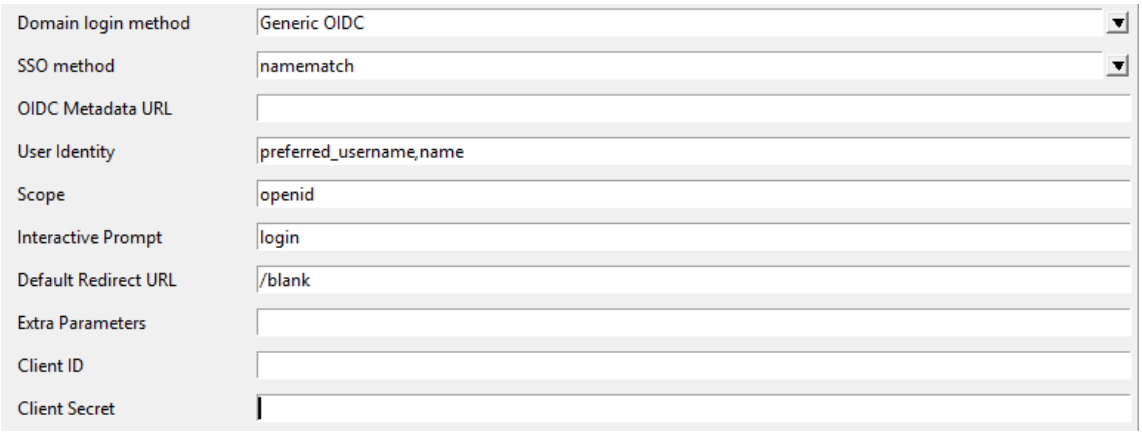
> **Note:**
>
> - 'http://' URIs are only permitted for 'localhost' for test purposes. If you need to test Okta on a Maconomy installation hosted on a different machine, you have to configure a TLS proxy for it.
> - **Do not** end specified redirect URIs with a '/'.
> - Always add the default redirect URI: 'https://{host-name}:{port}/blank' or 'http://localhost:{port}/blank'.

Configure Maconomy

To enable the OIDC login module and link Okta users to Maconomy users:

1. In MConfig, go to the OSGi Products window.
2. In the **Domain login method** field, select **Generic OIDC**.

   Once you select this option, the following fields display:



> **Note:** MConfig automatically fills out some of the fields with default values. Edit these if needed.

3. Fill out these mandatory fields.
   a. In the **SSO method** field, choose one of two options: **namematch** or **namemapping**.
   b. In the **OIDC Metadata URL** field, enter the following value: https://{account-name}.okta.com/.well-known/openid-configuration.
   c. In the **Extra Parameters** field, set the parameters **nonce**,**state**. The OIDC login module in the Coupling Service assigns them a unique ID and UUID, respectively.
   d. In the **Client ID** field, specify your OIDC client ID.
   e. In the **Client Secret** field, specify your OIDC client secret.
   f. Accept the default values provided for the rest of the fields.

> **Note:** Customers on Enterprise Cloud who want to activate Okta will need to send this information to Deltek Cloud Operations via a support case.

4. To save your changes, click **OK**.
5. In the succeeding dialogs that display, click **OK** or **Next** (whichever applies) to install your changes.
6. In the Workspace Client, go to the workspace that exposes the User Role Information container (for example, the Users workspace).
7. Link Okta users to Maconomy users. This depends on the value you selected for the **SSO method** field in the OSGi Products window of MConfig.
   - If you specified **namematch**, make sure the names of your Okta users and the names of your Maconomy users match.

- If you specified **namemapping**, enter the **user principal name (UPN)** or email address components of your OneLogin users ('user@domain') as Network Username and Domain Name in Maconomy.

Generic Support for Third-Party SSO Vendors Using OAuth 2.0 OIDC

Maconomy offers the capability for you to integrate with other third-party SSO vendor solutions that use the OAuth 2.0 OIDC protocol.

Note that there are no plans to support vendors utilizing the older OAuth 2.0 protocol, except when an SSO vendor implements OAuth 2.0 as a compatible subset of their OIDC offering. In such cases, Maconomy may be able to consume OAuth 2.0 Bearer tokens issued by the SSO vendor, in addition to the authorization code credentials issued by the OIDC protocol. For more information on this, see the section on OAuth 2.0 Bearer tokens further down.

Limitations: At present, this generic login support is only applicable to the Workspace Client and the web client. It does not include Touch, whose user interface requires tailoring to each authentication provider. It also does not include Enterprise Cloud, which only supports the Azure,OneLogin, and Okta authentification providers.

Note: While this generic login solution has been successfully tested with multiple third-party SSO vendors, there are no guarantees that other vendors out of the vast array that exist will not have additional requirements that are not supported at present.

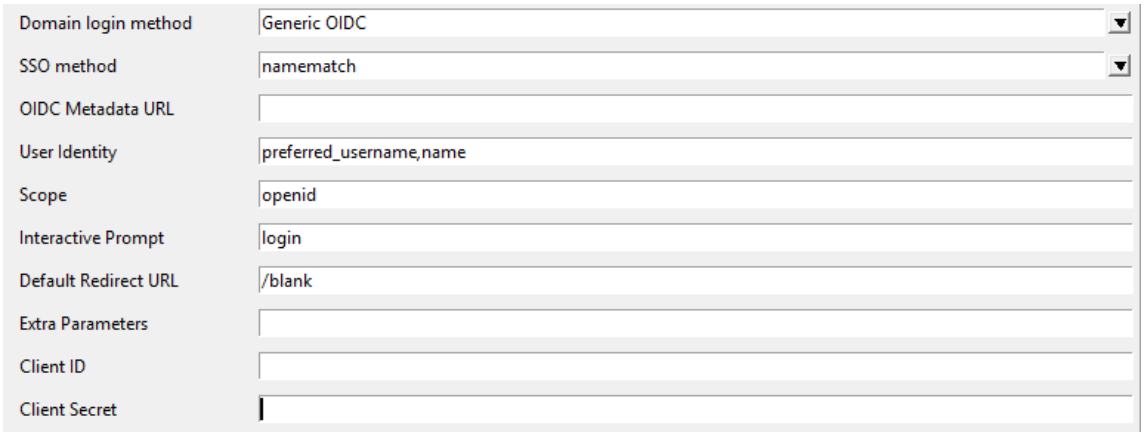
Setup Instructions

Configure Maconomy

To enable the OIDC login module:

1. In MConfig, go to the OSGi Products window.
2. In the **Domain login method** field, select **Generic OIDC**.

   Once you select this option, the following fields display:



Note: MConfig automatically fills out some of the fields with default values. Edit these if needed.

3. Fill out these fields. Except for **Extra Parameters**, all fields are mandatory.
   a. In the **SSO method** field, choose one of two options: **namematch** or **namemapping**.
   b. In the **OIDC Metadata URL** field, specify the metadata URL from which you want to obtain the OIDC provider configuration.
   c. In the **User Identity** field, enter the prioritized list of JSON Web Token (JWT) claims from which the OIDC login module will extract the user identity. Suggested values are:
      - **preferred_username,name** (if you specified **namematch** as the SSO method)
      - **upn,email** (if you specified **namemapping** as the SSO method)
   d. In the **Scope** field, enter a comma-separated list of scope values to which the OIDC login module will request access.
   e. In the **Interactive Prompt** field, enter the OIDC prompt value to use when initiating an interactive login. You can combine any of the standard OIDC 1.0 prompts.
   f. In the **Default Redirect URL** field, enter the default redirect URL to use for native clients (for example, the Workspace Client).
   g. You may also want to fill out the **Extra Parameters** field.
      - For example, you could specify a list of key=value assignments for inclusion in the authorization request. If you specify the nonce and state parameter keys without a value, the OIDC login module in the Coupling Service assigns them a unique ID and UUID, respectively.
      - If you want to generate such IDs for other parameter keys, set the relevant key's value to $id or $uuid. If a key starts with a plus sign, the OIDC login module appends the ID value to the existing value for the key. If a key starts with a minus sign, the OIDC login module removes the key from the authorization request.

   Note: For more information about which parameter values you should include, refer to the OAuth 2.0 OIDC documentation of the authentication provider.

   h. In the **Client ID** field, specify your OIDC client ID.
   i. In the **Client Secret** field, specify your OIDC client secret.
4. To save your changes, click **OK.**
5. In the succeeding dialogs that display, click **OK** or **Next** (whichever applies) to install your changes.

OAuth 2.0 Bearer Tokens

The OIDC protocol is designed to provide end-users with a flexible and integrated authentication experience. It is generally not suited for other authentication scenarios, such as scripted logins and integrations, where the security requirements might instead involve application access tokens, or even certificate-based authentication to entirely avoid having to store clear text credentials.

For such use cases the basic OAuth 2.0 protocol still serves a purpose. If the external SSO vendor offers an authentication endpoint through which a client can obtain an OAuth 2.0 Bearer token, the Maconomy RESTful API accepts this token when submitted in a 'Bearer' Authorization header, and passes it to the Maconomy OIDC login module for validation. If the token is a well-formed and signed JSON Web Token (JWT), the login module inspects the JWT for one of the user claims configured in the User Identity field in the Generic OIDC configuration. If a matching user claim is found, the login module looks for a Maconomy user with a matching Login Name or Network Username and Domain.

Note: It might be necessary to edit the User Identity field in the Generic OIDC configuration to include the user claim that contains the caller's identity in the OAuth 2.0 Bearer Token.

Set Up Passwords

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.

To ensure a secure system for custom implementation, MConfig auto-generates a password for the Administrator user and displays that at the end of the installation process. The initial password is composed of six random characters.

Password Requirements

Maconomy enforces password length over complexity. Note that no specific combination of characters is required.

Following are new password guidelines.

Password may contain:

- Up to 30 characters
- The ASCII characters [a..z] [A..Z] and digits [0..9]
- Special characters: ? * # / _ [ ] { } + - : . ~

Password may *not* contain:

- More than 30 characters
- Blank spaces
- Empty password
- A user name
- National characters, such as the Danish æ, ø and å

> **Note:** There is currently no minimum password length.

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:

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.

Use Password Reset

To use the password reset process:

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.

Two-Factor Authentication (2FA)

It is a Deltek-wide initiative to implement Two-Factor Authentication (2FA) in all Maconomy interfaces. Using 2FA reduces identity and access management risks for users. The Maconomy 2FA system provides an additional layer of authentication on top of the existing authentication mechanisms in order to decrease the risk of unauthorized access in the case where a user's credentials have been compromised.

The system is based on the standard Time-Based One-Time Password algorithm (TOTP) and is compatible with a range of smart device apps designed for this purpose, including but not limited to Duo Mobile, Google Authenticator, and Microsoft Authenticator. Push-based authentication is currently not supported.

Warning: The web client and Touch do not support the combination of Maconomy 2FA and Azure.

Maconomy 2FA Implementation

This section provides basic details on how a Maconomy user uses two-factor authentication to access Maconomy. The details and steps of the process vary slightly depending on the authenticator tool and the operating system you use. However, the main steps and interaction points are consistent across apps.

Some popular authentication apps include:

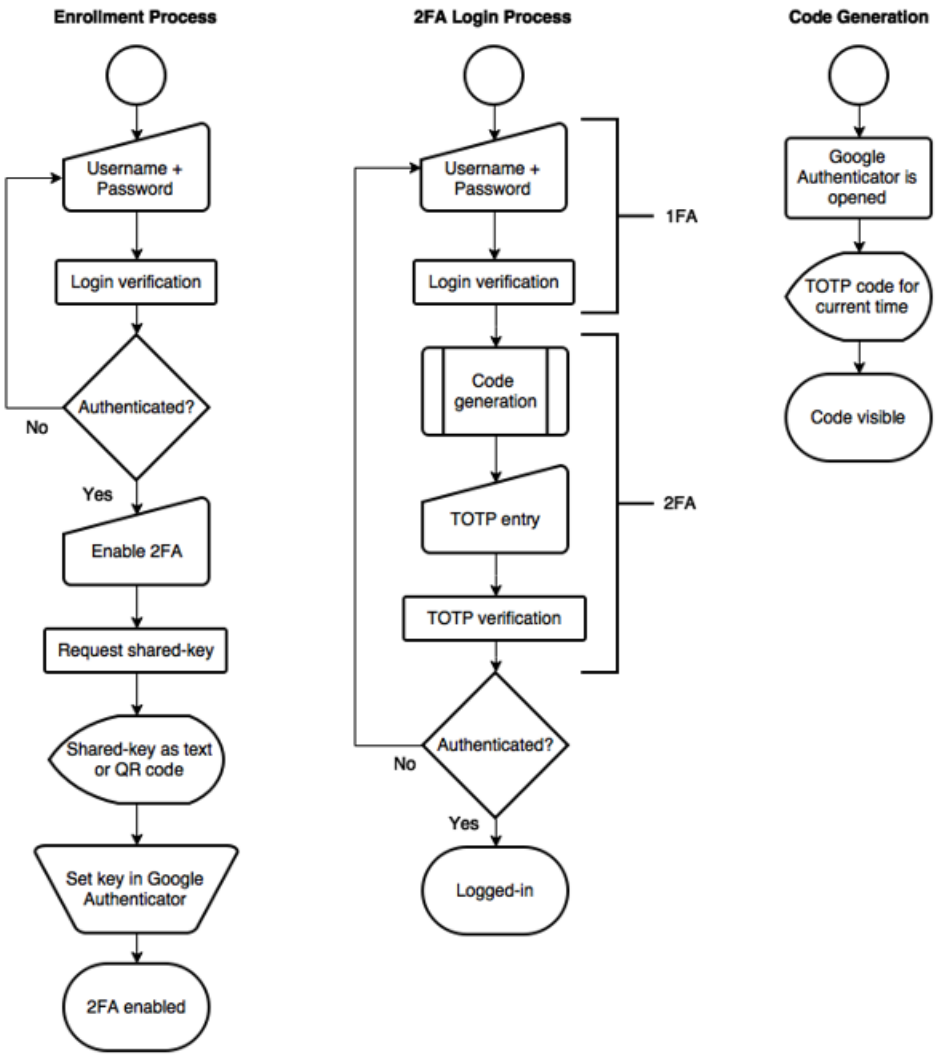
- Duo Mobile
- Google Authenticator
- Microsoft Authenticator

Process Summary

The first time a user logs in to Maconomy, he or she is asked to enroll a smart device app in the 2FA system by scanning a QR code. After the code has been scanned, the app is able to generate one-time passwords which consist of six digits and are valid for a limited period of time, typically 30 seconds. A one-time password must be provided by the user whenever the user's primary login credentials are required by the system. This works to prove that the user is in possession of the device on which the app is installed.

2FA Process Diagram

The following diagram represents the enrollment, login, and code generation processes:



2FA Enrollment and Reconfiguration

The Maconomy 2FA login module allows users to reconfigure their 2FA device.

To allow enrollment and reconfiguration:

Ensure the following are enabled in MConfig's OSGi product configuration:

- 2FA account reset — Enabled with the option **Allow users to reconfigured their 2FA device**.
- Reset password — Select **Enable "Reset password"** check box to enable.

If enabled, Maconomy 2FA login module supplies a temporary access token to allow users to reconfigure their 2FA device. The token is sent by email to the email address stored in the user's employee information. For added security in Maconomy 2.5.3 forward, users are also required to supply an access token sent by email to enroll their 2FA device for the first time.

To disable this feature:

If you wish to disable this security measure which requires users to supply an access token when they initially enroll their 2FA device, manually update the Maconomy 2FA login module parameter **accountResetToEnroll** to **false**.

Warning: When **accountResetToEnroll** is enabled, only users with a valid email address in their employee information are able to enroll in 2FA and gain access to Maconomy. Users without an email address are disqualified from 2FA, and must either use a login method which is skipped by the 2FA login module, or be assigned a **0** (zero) 2FA account, as described in the section 2FA Administration and Management below.

2FA Bypass Token

From a usability standpoint, it is not desirable to force the user to go through the Two-Factor Authentication process on every login. Once a user has proven ownership of a device by performing a successful 2FA login, it is not unreasonable to treat the device itself as a trusted second factor.

A device is trusted if it is in possession of a valid 2FA Bypass Token. This token is only returned to the device after a successful login involving Two-Factor Authentication has been completed. This token can then preemptively be included in subsequent login requests as a way to bypass any 2FA requirements for the user.

The 2FA Bypass Token is only valid for a specific Maconomy 2FA account, and for a server configured bypass period. When a valid 2FA Bypass Token is received by the server it will return an updated token with an extended bypass period.

By default 2FA Bypass Tokens are enabled for the Maconomy 2FA login module with a bypass period of 30 days. Bypass tokens are currently supported for the web client and Touch but not for the Workspace Client.

Authentication Keys

In order to provide fine-grained control over the behavior of the Maconomy TOTP login module, the concept of authentication keys was introduced. These keys are set by the other Maconomy-related login modules or by the login framework itself.

An authentication key is an assertion that a specific condition was met during the login process. For example, the 'CloudMFA' key indicates that a Cloud-based login with Multi-Factor Authentication (MFA) was performed, while 'WebLogin' is set if the login process was started from the Maconomy web interface (REST API).

By including some of these authentication keys in the 'skipIf' option list, you can control the circumstances under which 2FA should be skipped. The full list of supported authentication keys is shown in the table below:

Authentication Key	Description
Maconomy2FA	The login included Maconomy 2FA, or validation of a 2FA-bypass token
CloudLogin	A cloud login was performed
CloudMFA	A cloud login including MFA was performed
DomainSSO	A domain single sign-on was performed
PasswordLogin	A conventional password-login was performed
ReconnectLogin	A reconnect to an existing (but expired) login session was performed

Authentication Key	Description
WebLogin	A web interface (REST API) login was performed
WSCLogin	A Workspace Client login was performed

By default, 2FA is skipped for CloudMFA, DomainSSO, and ReconnectLogin.

TOTP Configuration

The core configuration of the TOTP authentication scheme is done in the main Coupling Service configuration file ('configuration/server.ini'):

```
server.auth.totp {
  issuer = issuer-name
  window-size = window-size
  algorithm = algorithm
  digits = digits
  time-step = time-step
}
```

Setting	Default Value	Description
issuer-name	Deltek Maconomy	The issuer name shown on the 2FA devices
window-size	2	The number of time-step windows to accept one-time passwords from. 2 means the current window + the previous window.
algorithm	SHA-1	The hashing algorithm to use. Changing this value from the default (SHA-1) might not be supported by all 2FA devices.
digits	6	The number of digits in the one-time password. Changing this from the default (6) might not be supported by all 2FA devices.
time-step	30	The duration (in seconds) of each time-step window. Changing this from the default (30) might not be supported by all 2FA devices.

2FA Administration and Management Capabilities

When a user completes the Maconomy 2FA setup process, a 2FA key is generated for the user. This key encodes both a TOTP key and a timestamp tracking the last time a one-time password was successfully validated for this account.

The 2FA key is stored in the Maconomy Key Store, which is introduced as safe storage for sensitive authentication-related information associated with each Maconomy user. The keys are stored in the key store with the key type 'Maconomy2FA'.

As an administrator, you can manage and interact with the Maconomy Key Store via several command-line options.

Key Store Interaction Examples

The following examples show how to interact with the Maconomy Key Store using the Maconomy Server command-line option '--keystore' which was introduced along with the Maconomy Key Store itself. In these examples, *MaconomyServer.app* is used as a placeholder for the MaconomyServer executable or symbolic link with the application name included (-i<app-name>) while *shortname* is a placeholder for the database shortname of the Maconomy application.

View all stored Maconomy 2FA key store entries:

```
$ MaconomyServer.app -Sshortname --keystore:type=Maconomy2FA
```

Create/update a Maconomy 2FA account. This is primarily useful if a user should always be exempt from 2FA which can be achieved by creating a 2FA account with key value '0' (zero):

```
$ MaconomyServer.app -Sshortname --keystore:type=Maconomy2FA:user=username:key=0:set
```

Delete a user's Maconomy 2FA account. When a 2FA account is deleted the 2FA setup process will be re-run the next time the user logs in:

```
$ MaconomyServer.app -Sshortname --keystore:type=Maconomy2FA:user=username:remove
```

> **Warning:** The 'remove' command deletes all keystore entries matching the entered criteria without asking for confirmation. Always run the command without 'remove' first to ensure the command only matches the desired key store entries.

MConfig Support for Maconomy 2FA

Setup

Maconomy 2FA can be enabled in MConfig during installation of Maconomy. You can choose whether users should be allowed to reconfigure their 2FA device in the event that it is lost or stolen. The reconfiguration process is based on the same email-token authentication scheme which can be enabled to allow users to reset their Maconomy password.

Advanced Setup

The Maconomy 2FA login process is configured via the login module 'MaconomyTOTPLoginModule' in the Coupling Service security configuration file ('configuration/maconomy.security.config'):

```
org.eclipse.equinox.security.auth.module.ExtensionLoginModule required
  extensionId="com.maconomy.lib.coupling.MaconomyTOTPLoginModule"
    QRCodeWindowHeight="window-height"
    QRCodeWindowWidth="window-width"
    allowAccountReset="allow-account-reset"
    skipIf="auth-key-list"
    ;
```

Setting	Default Value	Description
QRCodeWindowHeight	340	The suggested height (in pixels) of the window frame showing the QR code to the user.
QRCodeWindowWidth	340	The suggested width (in pixels) of the window frame showing the QR code to the user.

Setting	Default Value	Description
allowAccountReset	true (if “Reset password” is enabled)	Whether users should be allowed to reconfigure their 2FA device. Valid values are “true” and “false”.
accountResetToEnroll	true (if “Reset password” is enabled)	Whether users must provide an access token to enroll a 2FA device, like when they reconfigure their 2FA device.
skipIf	CloudMFA, DomainSSO, ReconnectLogin	A list with the authentication keys which will cause 2FA to be skipped for a login session.
allowBypass	true	Whether 2FA Bypass Tokens are enabled. Valid values are “true” and “false”.
bypassPeriod	30d (30 days)	The validity period of the bypass tokens, with time unit included. Valid time unit values are ‘d’ (days), ‘h’ (hours), ‘m’ (minutes) and ‘s’ (seconds).

Access Control

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**

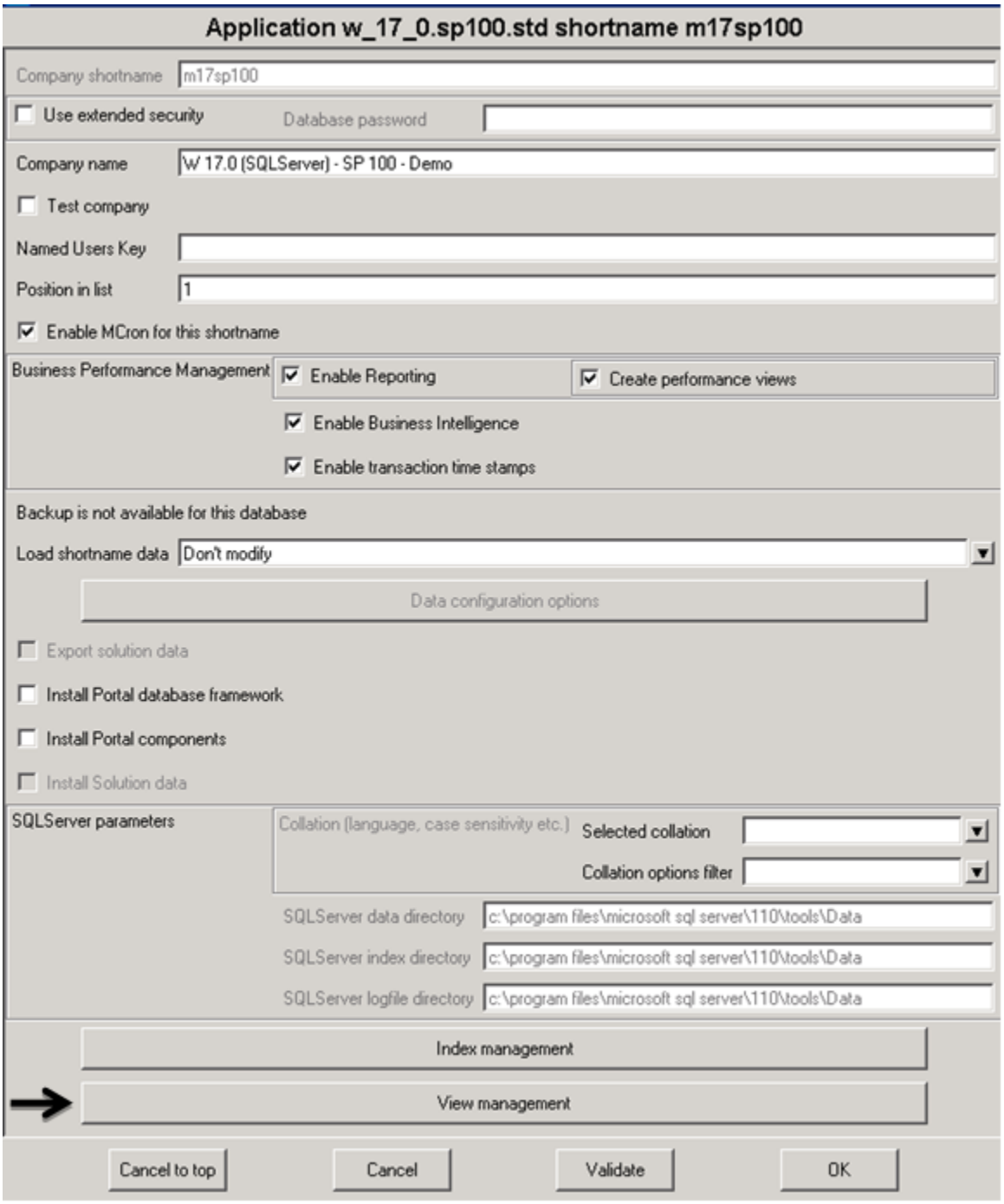
ACJOBENTRY

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			
default	1s	0s	0s
noemptykey	**1s**	**0s**	**0s**
union	180+s	180+s	180+s

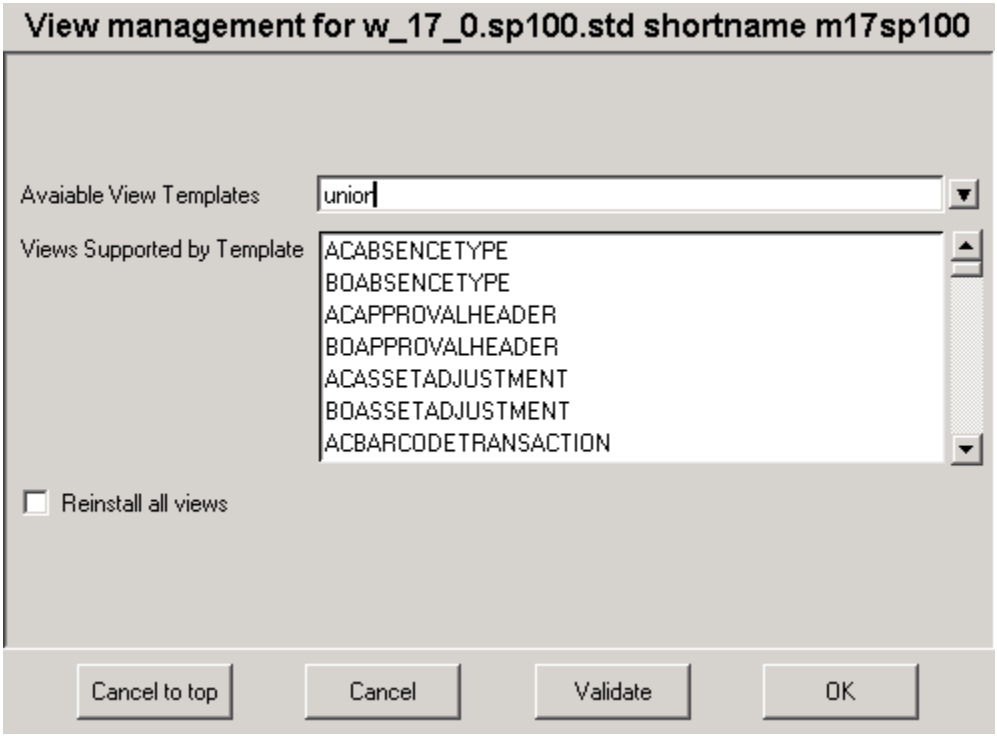
Setup

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.

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.

> **Note:** 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]
```

Install and Set Up Fonts

This section 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.

> **Note:** Google Noto Fonts are also available.

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.

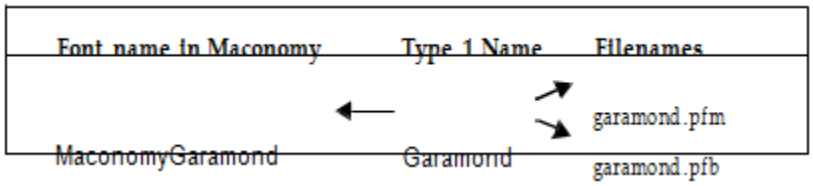
3. In the Fonts window, select **Install New Font** in the File menu and choose the drive and folder in which your new font was placed.
4. Click **Select All**.
5. Make sure that the **Copy fonts to Fonts folder** check box is selected.
6. Click **OK**.
7. In the progress dialog, click **Yes** for all 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.”

Move MPL into Coupling Service

Moving MPL into Coupling Service

Phase 1 moving MPL into Coupling Service supports large printings which had previously a run out of memory.

The MPL print engine in Maconomy is ported to run inside the Coupling Service, instead of as an embedded Java process inside each MaconomyServer intance.

This reduces the average memory footprint of the MaconomyServer processes, while also making it possible to produce larger prints without running into memory limitations.

The embedded MPL engine is still available in the 2.5 release stream in order to support various command line operations, such as the '-U*P' arguments as well as MScript and import invocations, which require print functionality. In future releases, the embedded print engine will be removed entirely, and

these operations will either be ported to the CouplingService, or they will be desupported. The '-U*P' arguments as well as the '-x' argument for stand-alone script execution are supported by the CouplingService as of 2.5.1.

QR Code Centre Image Support

MPL supposrts QR Code Centre Image Support to fulfill statutory e-invoicing requirement.

You can overlay images at the center of such QR codes without compromising the QR code's readability. A QR code scanner/app is now able to scan such a QR code successfully even though part of the QR code area is overlaid by an image.

For example, the following code:

```
<barcode {"Hello world"}
    QRCODE
    height=46mm
    qrOverlayImage={"qrcode\image.png"}
```

will overlay the qrcode\image.png from the Maconomy's document archive at the center of the QR code.

Technically, overlaying images over QR codes is possible because QR codes implement error correction, which means that a QR code can still be read by a scanner even though part of it is damaged or covered by an image. The QR code specification lists the following error correction levels and their corresponding theoretical recovery rates (the percent of the coding words that can be missing without affecting the QR code's readability).

- L: 7%
- M: 15%
- Q: 25&
- L: 30%

In practice, for the QR codes to scan correctly, the overlay should often take up less than that theoretical limit. It is up to the MPL layout developer to choose the right error correction level and overlay image size depending on the use case. The chosen values should be tested extensively in practice.

These two properties can be set using the following <barcode> attributes:

- qrErrorCorrection - the requested error correction level. Valid values are: L (lowest), M, Q, H (highest).
  The default values are
  - L without an overlay image
  - M when an overlay image
- qrOverlayAreaRatio - the ratio of the overall qr code picture's area (including the QR code's silent zone) that the overlay picture should take up. Valid values are from 0.0 to 1.0.
  The default value for error correction level M is 0.05.

Apart from these two attributes, you can also set the requested overlay image quality with the qrOverlayQuality attribute. Since the overlay image is drawn over the QR code image, the QR code image must be scaled up to have enough pixels to draw the overlay image over without losing quality.

qrOverlayQuality - the requested quality level. The valid values are: L (lowest), M, Q, H (highest). Level H indicates that the overlay image should be drawn with original quality and the QR code image should be scaled up accordingly (up to 2000 x 2000 pixels max). The higher the quality level, the more RAM

memory the QR code image will consume when generating a PDF and the bigger the resulting PDF size will be. It is recommended that you use the lowest possible value that renders a satisfying quality.

The default value is Q.

The following examples shows how to set all the above described attributes:

```
<barcode {"Hello world"}
   QRCODE
   height=46mm
   width=46mm
   qrOverlayImage={"qrcode\swiss_flag.png"}
   qrErrorCorrection=M
   qrOverlayAreaRatio=0.025
   qrOverlayQuality=L>
```

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 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
```

[2] The standard Type 1 fonts known to the system are *Helvetica*, *Times-Roman*, *Courier*, *Symbol* and *ZapfDingbats*.

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 `F`ONTWIDTHSCALEFACTOR 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) 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. It can either be a single font, or a comma separated list of fonts. When a list of fonts is used, the MPL 4 engine selects the appropriate font for each character in the printout, such as the first font in the list to support the character.

```
[MPLConfig]

DefaultFont=HelveticaUni

#or

DefaultFont= HelveticaUni, JhengHei, Leelawade
```

> **Note:** It is important to recompile all standard and custom layouts after changing the default font. To recompile the standard layouts, use the MaconomyServer -UP command. Please refer to the Application Server section for a detailed description of MaconomyServer commands and the -UP option in particular.

The MaconomyServer -UP command has the following shape:

```
MaconomyServer.<APP_NAME>.cmd -S<shortname> -UEP
```

For example:

```
MaconomyServer.w_20_0.cmd -Sw200 -UP
```

Similarly, in order to recompile all the custom print layouts after the default font setting has been changed, use the MaconomyServer -UEP to first export the custom print layouts and MaconomyServer -UIP to import them back.

For example:

```
MaconomyServer.w_20_0.cmd -Sw200 -UEP

MaconomyServer.w_20_0.cmd -Sw200 -UIP
```

Defining Last Resort Font in MPL4

In MPL 4, you can define a Last Resort Font to use for rendering characters that are not supported by any of the configured fonts (either default fonts or the current value of the MPL's fontname attribute). The Last Resort Font should render appropriate "missing character" glyphs for any unsupported characters.

Maconomy pre-packages the official Unicode Last Resort font (https://unicode.org/policies/lastresortfont_eula.html) into the TPU, under FontSupport\fonts\LastResort.ttf. The Unicode Last Resort font covers the entire world and renders an appropriate “missing character” glyph for any Unicode language family. For example, the following glyph is rendered when a Simplified Chinese character is encountered, but no font supporting Simplified Chinese (which is part of the CJK language family) is configured for Maconomy.

To configure the Last Resort Font, edit the following section of your custom Maconomy.ini file to point to the chosen font.

```
[MPLConfig]

LastResortFont=LastResort
```

In addition that that, make sure that the FontSupport\FontFileMap.lst file contains the name of the font followed by a tab character, then followed by the name of the LastResort font file in the FontSupport\fonts directory, for example:

```
LastResort    LastResort.ttf
```

The FontSupport\FontAttributeMap.lst must contain the tab separated name of the font, for example:

```
LastResort    LastResort
```

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
   ```

6. Recompile all Standard and Custom print layouts by running the following Maconomy Server commands:
    - -UP, e.g.: `MaconomyServer.w_20_0.cmd -Sw200 –UP`
    - -UEP, e.g.: `MaconomyServer.w_20_0.cmd -Sw200 –UEP`
    - -UIP, e.g.: `MaconomyServer.w_20_0.cmd -Sw200 –UIP`

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
```

Remember to recompile all the standard and custom print layouts after setting the default font to the new value.

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.

Google Noto Fonts

Google's font family, called Noto, supports all languages with a similar style. Maconomy includes four Google Noto fonts, and from Google customers can obtain additional fonts for broader language needs.

From Google: Noto has multiple styles and weights, and is freely available to all. The comprehensive set of fonts and tools used in our development is available in our GitHub repositories.

To see more information, go to:

https://www.google.com/get/noto/

Note: Noto is a trademark of Google Inc. Noto fonts are open source. All Noto fonts are published under the SIL Open Font License, Version 1.1. Language data and some sample texts are from the Unicode CLDR project.

Printing Right-to-Left Languages

Maconomy supports the entry and display of data in right-to-left languages, and the Maconomy Print Language (MPL) supports the correct printing of such text. Specifically, companies can:

- Configure Maconomy to print data in right-to-left languages in an otherwise left-to-right language print. For example, you can print Arabic customer names within an English-language print.
- Customize MPL layouts so that static content on a print is in a right-to-left language as well. For example, you can create a custom invoice entirely in Arabic.

For the vast majority of use cases, the **DefaultTextDirection** configuration setting**,** editable in the MaconomyCustom.ini server config file, should be set to LTR (left-to-right bi-directional printing). Suppose English is an example left-to-right language and Arabic is an example right-to-left language. Setting **DefaultTextDirection** to LTR will cover the correct printing of English-only language text, Arabic-only language text, and instances where Arabic appears within English text.

For specific instances where text is predominantly Arabic but contains some English phrases, read further to understand options to support this.

The following sections provide a conceptual overview of bi-directional printing, and setup instructions for enabling this functionality. These setup instructions should apply for the majority of companies and their users.

Bi-Directional Printing

The **textdirection** MPL style attribute supports the one-directional left-to-right mode of printing, as well as the bi-directional modes of printing which are both capable of printing right-to-left languages correctly.

The succeeding paragraphs use English as the left-to-right language, and Arabic as the right-to-left language. The concepts discussed should apply to all left-to-right and right-to-left languages, respectively.

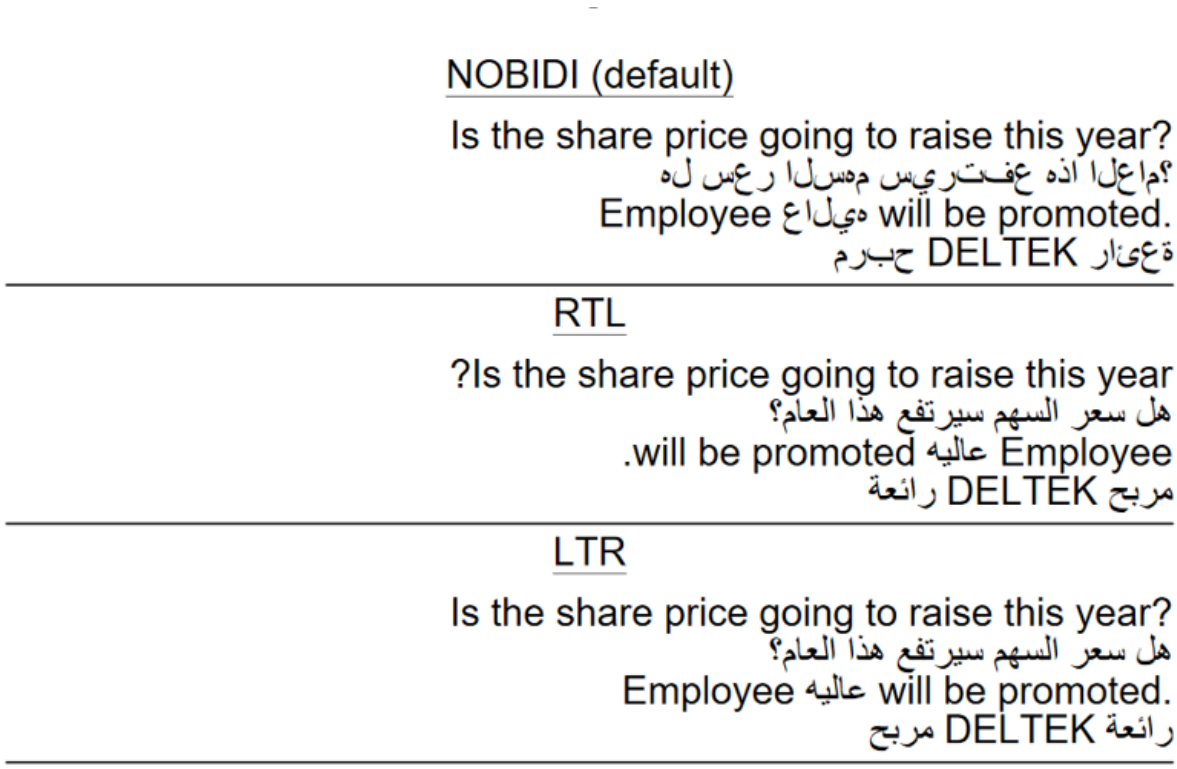
The **textdirection** attribute can assume any of the following values:

- **NOBIDI** – The one-directional (or non bi-directional) default that is equivalent to how MPL used to work prior to the introduction of bi-directional printing. Text is always printed left to right independently of the language used.
- **RTL** – Uses bi-directional reordering with left-to-right preferential run direction. With this algorithm, the text is divided into English and Arabic blocks. Characters comprising Arabic blocks are always printed right to left. Moreover, the blocks themselves are printed in the **right to left** order. This setting is ideal for embedding English within Arabic text, where all text reads right to left apart from English words which read left to right.
- **LTR** – Uses bi-directional reordering with right-to-left preferential run direction. With this algorithm, the text is also divided into English and Arabic blocks. Characters comprising Arabic blocks are always printed right to left. Moreover, the blocks themselves are printed in the **left to right** order. This setting is ideal for embedding Arabic within English text, where all text reads left to right apart from Arabic words which read right to left.

To better understand these settings, consider the following sample sentences:

1. Is the share price going to raise this year?
2. هل سعر السهم سيرتفع هذا العام؟ *[English translation of 1]*
3. Employee عاليهه will be promoted. *[English: Employee* Aaliyah *will be promoted.]*
4. مربح DELTEK رائعة *[English: Company DELTEK is profitable]*

The following screenshot shows how these sentences will be printed in MPL using the three settings.



Note that when using LTR, sentences 1, 2 and 3 are printed correctly, but 4 is incorrect. On the other hand, when using RTL sentences 2 and 4 are correct, and 1 and 3 incorrect. NOBIDI is not suitable for printing Arabic at all, as it prints everything left to right.

To conclude: When using Arabic words in the English context, the LTR textdirection is appropriate. For embedding English words in the Arabic context, RTL is a better fit.

Setting Text Direction

You can use the **DefaultTextDirection** attribute to specify the system-wide Maconomy setting. To do this, update the MaconomyDir/Definitions/MaconomyCustom.ini server config file.

For example:

```
[MPLConfig]
DefaultTextDirection=LTR
```

Setup Instructions

For purposes of backward compatibility, bi-directional printing is disabled by default.

Once enabled, this functionality works for both the Workspace and web clients. .

> **Note:** If you are a customer on Maconomy Cloud who needs to utilize this functionality, file a support case with Deltek Cloud Operations.

To enable this feature:

1. Make sure you have installed the right fonts to render your right-to-left language (if applicable). For more information, see the Font Administration and Installation in MPL 4 section.
2. By default, the system-wide text direction is set to **NOBIDI**. To configure text direction for your entire system:
    a. In the MaconomyDir/Definitions/MaconomyCustom.ini file, specify a new value for DefaultTextDirection.

    For example:

    ```
    [MPLConfig]
    DefaultTextDirection=LTR
    ```

    b. Recompile all standard and custom print layouts.
        i. To recompile the standard layouts, use the `MaconomyServer –UP` command.
        ii. To recompile your custom print layouts, use the `MaconomyServer –UVP` command.

    See the Maconomy Server Executable Options section for a detailed description of MaconomyServer commands and the –UP/–UVP options in particular.

    For example:

    ```
    MaconomyServer.w_21_0.cmd-Sw210 –UP
    MaconomyServer.w_21_0.cmd-Sw210 –UVP
    ```

Install and Upgrade

Automated Hotfix (CU) Installation

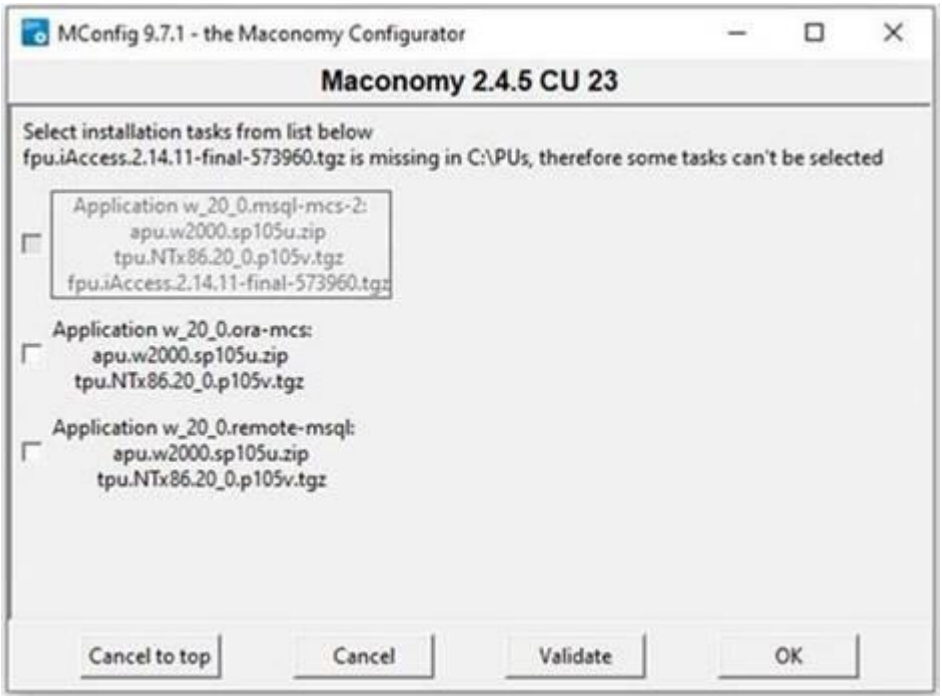
An installer specialized for Cumulative Update (CU) installations simplifies the process of installing CUs. The CU installer, which is a hardcoded MConfig installer applying the same underlying functionality as MConfig, handles APUs, TPUs, web client PUs, and standard SPUs.

Note: Each CU installer is built for a specific CU, and can be used only for that particular CU. For example, a CU installer for Maconomy 2.5.1 CU 5 can be used for only for installing 2.5.1 CU 5, not others, such as 2.5.1 CU 4.

To use the CU installer:

1. Manually move the relevant PUs to the packing units folder on the machine. All relevant PUs are automatically installed by the installer.
2. Ensure that all the PUs are in the packing units folder. This means that if a CU includes both an APU and a TPU, you cannot choose to install the TPU only. In this example, you must include both the APU and TPU.
3. In the CU Installer folder, right-click the **Install[version].exe** installer and select **Run as Administrator**.
4. The Maconomy Configurator screen displays, prompting you to select the needed installation tasks. The installer scans the list of applications on the server and finds which applications the CU is relevant.

   Select the applications for which the CU should be installed and then click **OK**.



   In the screenshot example above, the CU can be installed on the applications w_20_0.ora-mcs and w_20_0.remote-msql. The CU could have been applied to w_20_0.msql-mcs-2 as well if the web client PU had been present.

5. Select the applications to install and click **OK**.

6. Enter the database system password (as you do when using MConfig). The MConfig Scripting Console runs automatically.
7. If the application uses remote database servers or remote web servers, start the wiserver program on the remote machines. Click **OK**.

   The installer installs the CU on the selected application(s) and closes down.

Windows Installation

Deltek Best Practice

Windows Installation

From DSM, download the latest Maconomy Workspace Client installation package.

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.

Warning: 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 the latest Maconomy Workspace Client installation 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.

Warning: 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.

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 can impact customer networks, including information that updates are at present passive until users actively attempt to open the WSC following a new change. Maconomy comes with the ability to configure a script to push updates more efficiently to users on the network whenever possible, 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.

By default, the Workspace Client displays a warning message to inform you when an update is available. To disable this and allow the system to automatically update the system, you can edit the CS server.ini file.

To do so, edit the following property in the CS server.ini file:

```
client.p2.showInstallAndUpdateWarning = <boolean>
```

The default value is **true**. Change it to **false** to remove the warning message.

Silent Updates

You can run a command to enable Maconomy to push updates whenever possible, instead of when users actively open the WSC.

In Windows, open the command prompt and run the following command:

```
Maconomy.exe --silentInstallAndUpdate -a <host> -p <port>
```

Wait for the command to finish.

You can also run this command in MAC OSX. Open the Terminal and run the following command:

```
Maconomy.app/Contents/MacOS/Maconomy --silentInstallAndUpdate -a <host> -p
<port>
```

- Wait for the command to finish.

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.

Export to Excel Row Summary

The Export to Excel action enables you to include a count of the number of rows included in the generated file.

Administrators must configure this option prior to use. The following parameters are added to export-to-excel action ExportDataSet:

- **RowId (boolean)**—If enabled a row-id (number starting from 1) is added as the first column for all rows in the export. The color-format of the row-id column is identically to the header-color-code.
- **RowCount (boolean)**—If enabled a count (number) of all data-rows is added as the last line in the export. For example: Row Count: 1001

The system checks that the number of rows does not exceed 1048576, which is the maximum number of rows supported by Excel (including header and tail-info / row count).

Note: If the number of lines exceeds this number, the export is truncated.

Set Up Maconomy

Background Task Setup

See the *Deltek Maconomy Background Task Guide* for details.

User Masking Setup

To enable user, employee, and company masking you must set the value of the **Allow User Masking for Company** system parameter.

To enable masking, complete the following steps:

1. Navigate to **Setup » System Setup » Parameters and Numbers » System Parameters** tab.
2. Select **All Parameters** in the **Show** field.

   The table displays the list of all system parameters.
3. In the **Description** column, scroll to **Allow User Masking for Company** and double-click to open the **System Parameter** tab for that parameter.
4. Select **Allow User Masking for Company** in the **System Parameter** island in the **System Parameter** tab. The default value is false (not selected).
5. Enter any appropriate remarks in the **Remarks** field. Remarks are not required.
6. Enter or search for a company number in the **Company No.** column in the **Company Specific Values** sub-tab.

   This column is validated; the company number must exist.
7. Select the **Allow User Masking for Company** column to enable masking for this company. The default value is false (not selected).
8. Enter any appropriate remarks in the **Remark 1 – 3** columns. Remarks are not required.

Repeat this process to enter the company numbers for all of the companies for which you want to enable masking.

Long Text Support

Originally introduced in version 2.1.1, the Long Text functionality allows you to add a text extension to time sheet lines so that end users can enter descriptions longer than the 255-character limit. Maconomy also displays the long text field during invoice editing and before printing of the final invoice. End users can edit the description before printing the final invoice.

After a time sheet is approved, Maconomy transfers the long text references to the job entries. On the job entry, the long text is read-only. This means any workspaces that display job entries can display the corresponding non-editable long text. If a job is transferred or reallocated, the text reference is transferred to the new job entry.

Beginning with version 2.4.4, this functionality is available in the Workspace Client, web client, and Touch. In the Workspace Client, the Texts single dialog workspace displays all long texts used in the system, with the following information provided for each:

- Reference instance key
- Read-only status
- Whether the long text is in use

Note: These fields are not available by default in the standard layout, but in the PSO and CPA Solution layouts.

Setup Instructions

To allow long text entry, enable the **Enable Long Text Emulation in Workspaces** system parameter. Enabling this parameter activates the layouts that contain the long text field.

By default, this parameter is disabled.

Warning: If you enable/disable the system parameter after deploying the system to users, your standard layouts will stop displaying existing descriptions. As a temporary solution, you can utilize custom layouts for viewing these descriptions.

This limitation applies to all clients.

Containers for Long Text Support

Workspaces that utilize long text are based on containers that have namespace = "MaconomyLongText". The following table lists these containers and the names of the applicable fields' long text equivalent:

Container	Field/s
MaconomyLongText:InvoiceEditing	InvoicePrintoutLine.LongText (adjacent to existing InvoicePrintoutLine.Text)
MaconomyLongText:QuoteEditing	QuotationPrintoutLine.LongText (adjacent to existing QuotationPrintoutLine.Text)
MaconomyLongText:QuoteRevisions	QuotationPrintoutLine.LongText (adjacent to existing QuotationPrintoutLine.Text)
MaconomyLongText:Requisitions	RequisitionLine.LongLineText (adjacent to existing RequisitionLine.LineText)

Container	Field/s
MaconomyLongText:RequestsForQuote	RequestForQuoteLine.LongLineText (adjacent to existing RequestForQuoteLine.LineText)
MaconomyLongText:PurchaseOrders	PurchaseOrderLine.LongLineText (adjacent to existing PurchaseOrderLine.LineText)
MaconomyLongText:InvoiceAllocation	VendorInvoiceJournal.EntryText
MaconomyLongText:InvoiceReallocation (up to and including Maconomy 20 only)	VendorInvoiceJournal.EntryText
MaconomyLongText:ShowJobInvoice	InvoicePrintoutLine.LongText (adjacent to existing InvoicePrintoutLine.Text)
MaconomyLongText:ShowJobCreditMemo	InvoicePrintoutLine.LongText (adjacent to existing InvoicePrintoutLine.Text)
MaconomyLongText:JobInvoiceCrediting	InvoicePrintoutLine.LongText (adjacent to existing InvoicePrintoutLine.Text)

There are also Workspace Client containers that include the LongDescriptionDay1-7 and LongDescriptionWeek long text fields as an extension to the normal fields used in pane layouts. The following table lists these containers and the names of the applicable long text fields:

Container	Field/s
MaconomyLongText:TimeRegistrationDetails	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7) ▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:TimeRegistrationDetails DayDescription	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2)

Container	Field/s
	▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7)
MaconomyLongText:TimeRegistrationDetailsWeekDescription	▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:TimeSheetLines	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7) ▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:TimeSheetLinesDayDescription	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6)

Container	Field/s
	▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7)
MaconomyLongText:TimeSheetLinesWeekDescription	▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:DailyTimeSheetLines	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay) ▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:DailyTimeSheetLinesDayDescription	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay)
MaconomyLongText:DailyTimeSheetLinesWeekDescription	▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:TimeRegistration	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7) ▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:TimeRegistrationDayDescription	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4)

Container	Field/s
	▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7)
MaconomyLongText:TimeRegistrationWeekDescription	▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:ApprovalTimeSheets	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7) ▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:ApprovalTimeSheetsDayDescription	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7)
MaconomyLongText:ApprovalTimeSheetsWeekDescription	▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)

Container	Field/s
MaconomyLongText:ApproveTimeSheetLines	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7) ▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:ApproveTimeSheetLines DayDescription	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7)
MaconomyLongText:ApproveTimeSheetLines WeekDescription	▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:TimeSheets	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4)

Container	Field/s
	▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7) ▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:TimeSheetsDayDescription	▪ TimeSheetLine.LongDescriptionDay1 (adjacent to existing TimeSheetLine.DescriptionDay1) ▪ TimeSheetLine.LongDescriptionDay2 (adjacent to existing TimeSheetLine.DescriptionDay2) ▪ TimeSheetLine.LongDescriptionDay3 (adjacent to existing TimeSheetLine.DescriptionDay3) ▪ TimeSheetLine.LongDescriptionDay4 (adjacent to existing TimeSheetLine.DescriptionDay4) ▪ TimeSheetLine.LongDescriptionDay5 (adjacent to existing TimeSheetLine.DescriptionDay5) ▪ TimeSheetLine.LongDescriptionDay6 (adjacent to existing TimeSheetLine.DescriptionDay6) ▪ TimeSheetLine.LongDescriptionDay7 (adjacent to existing TimeSheetLine.DescriptionDay7)
MaconomyLongText:TimeSheetsWeekDescription	▪ TimeSheetLine.LongDescriptionWeek (adjacent to existing TimeSheetLine.EntryText)
MaconomyLongText:DailyTimeRegistration	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay) ▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:DailyTimeRegistrationDayDescription	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay)
MaconomyLongText:DailyTimeRegistrationWeekDescription	▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:DailyTimeSheets	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay)

Container	Field/s
	▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:DailyTimeSheetsDayDescription	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay)
MaconomyLongText:DailyTimeSheetsWeekDescription	▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:JobDailyTimeRegistration	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay) ▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)
MaconomyLongText:JobDailyTimeRegistrationDayDescription	▪ DailyTimeSheetLine.LongDescriptionDay (adjacent to existing DailyTimeSheetLine.DescriptionDay)
MaconomyLongText:JobDailyTimeRegistrationWeekDescription	▪ DailyTimeSheetLine.LongDescriptionWeek (adjacent to existing DailyTimeSheetLine.EntryText)

The containers postfixed ...DayDescription or ...WeekDescription only add said long text fields. These containers support only one long text (instead of several), and perform better than the general container.

Long Text in Invoice Editing

To display long text during invoice editing:

- The invoice layout rule should specify that invoices must print a line per registration.
- The Registration Specification – Field 1 detail must be either the DailyDescriptionTextHeaderInstanceKey field or the EntryTextTextHeaderInstanceKey field.

Note: Even when long text is enabled, users cannot enter descriptions exceeding 255 characters in header-type Description fields on all invoice lines. These lines have specific printing restrictions which make them unable to accommodate long text.

This exception applies to all clients.

Printing Considerations

Utilizing Long Text in Invoice Prints

To utilize the long text daily descriptions from time sheet lines in invoice prints, select **One Entry per** Day in the **Allocation Method** field in the System Information Workspace.

Splitting Long Text in Prints

For the workspaces that correspond to the containers listed above, the long text is stored as a series of records in the database. When printed, these records appear as continuous long text.

However, as the print area is limited, the chunks of text stored in each record must not exceed the space available in the corresponding print. In order to know where to split a long text, the system needs to determine the following:

1. What is the name of the font used for printing?
2. What is the font-size used?
3. Is the font boldfaced, italicized, and/or underlined?
4. What is the width available for printing (measured in points)? A point corresponds to 1/72 of an inch.

The configuration of these properties are stored in dedicated "option lists" in the database. To look up the properties for a specific area, use any of the following option lists:

- config:longtext:Invoice - used for Invoice Editing
- config:longtext:Quote - used for Quote Editing and Quote Revisions
- config:longtext:Requisition - used for Requisitions
- config:longtext:RequestForQuote - used for Request for Quotes
- config:longtext:PurchaseOrder - used for Purchase Orders
- config:longtext:InvoiceAllocation - used for Invoice Allocation and Invoice Reallocation

For each of these option lists, you can use the **Name** field for the property name, and the **Description** field for the corresponding value.

You can specify the following properties:

- fontName - the name of the font
- fontSize - the size of the font (in points)
- fontBoldface - true or false. True if the font is boldfaced.
- fontItalic - true or false. True if the font is italicized.
- fontUnderline - true or false. True if the font is underlined.
- width - a number indicating the available printing width (in points)
- indentation - a number indicating a possible indentation. For example, if the width = 200 and the indentation = 10, the available space is calculated as 200-10 = 190.

If you do not define any of these properties, the system uses default values for them. These default values roughly correspond to the factory settings of Maconomy prints.

Company-Specific Settings

In addition, you can opt for company-specific settings to address instances when one company uses layouts different from another company within a multi-company setup. To indicate settings specific to a company, create an option list with the "config"-option-list names listed above and with ".company" appended to each name, followed by the value of the company number.

For example, if you need an option list that indicates the invoice settings for a company assigned company number “1”, use the following format:

config:longtext:Invoice.company1

If you do not define any company-specific options lists, the system uses the general (non-company-specific) option list.

Usually, when a company-specific option list is specified, only that list is taken into account. Any property not specified for that list will instead use the default (factory-setting) value.

However, you can specify that the remaining values should instead be taken from the general (non-company-specific) option list. You do that by specifying a line in the option list with name = "addendum" and the corresponding description as true. This points Maconomy to the company-specific list as an "addendum" to the general list, rather than as a "stand-alone" list.

Line-Specific Properties

For some prints, the properties may differ for each specific line within the print. For example, in the standard invoice print, the indentation differs depending on the specified indentation level on the line. Second, the font differs depending on the value of the property header. Finally, the print settings might be different in invoices and credit memos. To accommodate such needs, you can further refine the properties. If you postfix certain specifications to a property, you can override general properties depending on the specific line that is printed.

For example, if you need to set different values for the “indentation” property of some instances of the **Indentation Level** field, postfix a given property with ".indentLevel0" for indentation level 0, ".indentLevel1" for indentation level 1, and so on.

That is, if you set the value for "indentation.indentLevel2" as “10”, and set the value for "indentation.indentLevel3" as “20”, lines with an indentation level of 2 are indented by 10 points, and lines with an indentation level of 3 are indented by 20 points.

You can specify properties that relate to whether a line is a header. When you do this, make sure you take the indentation level of the line into account. The postfix properties for header lines are ".header" for header lines and ".noHeader" for non-header lines.

Hence, if you want to declare that headers at indentation level 1 are boldfaced, specify the "fontBoldface.header.indentLevel1" property, and give it the value “true”.

Finally, you can refine the values for invoices and credit memos. When you do this, take the header/noHeader into account as well as the indentation level. The postfixes for invoices and credit memos are ".invoice" and ".creditMemo", respectively.

To specify a width of 230 points for credit memos and 200 points for invoices, specify the "width" as 200 and then specify all relevant width combinations of the width property for credit memos as:

"width.creditMemo.header.indentLevel0" (200)

"width.creditMemo.header.indentLevel1" (200)

"width.creditMemo.header.indentLevel2" (200)

"width.creditMemo.header.indentLevel3" (200)

"width.creditMemo.header.indentLevel4" (200)

"width.creditMemo.header.indentLevel5" (200)

"width.creditMemo.header.indentLevel6" (200)

"width.creditMemo.header.indentLevel7" (200)

"width.creditMemo.noHeader.indentLevel0" (200)

"width.creditMemo.noHeader.indentLevel1" (200)

"width.creditMemo.noHeader.indentLevel2" (200)

"width.creditMemo.noHeader.indentLevel3" (200)

"width.creditMemo.noHeader.indentLevel4" (200)

"width.creditMemo.noHeader.indentLevel5" (200)

"width.creditMemo.noHeader.indentLevel6" (200)

"width.creditMemo.noHeader.indentLevel7" (200)

Maconomy prioritizes and uses the most specific property. That is, it looks for and examines the properties in the following order:

1. Properties that contain invoice/creditMemo, header/noHeader, indentLevelN
2. If no property fits the first criterion: Properties that contain invoice/creditMemo and indentLevelN
3. If no property fits the first and second criteria: Properties that contain indentlevelN
4. If no property fits the first, second, and third criteria: “Pure” property

A similar but less complex way of having different properties based on the line-type is also available in the three purchase-related areas: Requisitions, Requests for Quotes, and Purchase Orders.

Although these three areas can have properties of their own, you would usually want to keep them completely identical. This relates to the fact that you can create a Request for Quote based on a Requisition, and a Purchase Order from a Request for Quote. You can even create a Purchase Order directly from a Requisition.

For these purchase areas, you can postfix a specific property based on the type of line. If fixed asset-related lines use a different font than job/cost-related lines, you can take that into account. The possible postfixes are:

- ".fixedAsset" for fixed-asset related lines
- ".inventory" for inventory-related lines
- ".jobCost" for job-cost-related lines
- ".g/l" for G/L-related lines
- ".subcontractor" for sub-contractor-related lines

To display and print sub-contractor-related lines in italics, specify the "fontItalic.subcontractor" property and give it the value "true."

Debugging

The Texts single dialog workspace (Single Dialogs » Set-Up » Note » Texts) lists all long texts in the system. This is useful for debugging purposes.

Advanced Logging Setup

This section contains setup information for Advanced Logging, and examples. See the Advanced Logging section for conceptual information.

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,
                            CustomerPaymentMode);

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	UpdatingLogNumber	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	20,00	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.

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

Synchronous Notification Setup

Note: The steps below are valid only for 2.5.1.

To set up Synchronous Notifications:

1. In MNSL, locate the asynchronous notification code, and delete the query definition (line highlighted in yellow) below.

```
<?xml version="1.0" encoding="UTF-8"?>
<MNSL xmlns="http://www.deltek.com/ns/mnsl" version="0.2">
  <Notifications name="ApprovePurchaseOrderLine" title="Approve Purchase Order Line">
    <Query entity="NotificationUniverses::Basics::ApprovePurchaseOrderLine">
      <Restriction condition="ApprovalLine.Approver = userEmployeeNumber() and userEmployeeNumber() != ''"/>
    </Query>

    <Link workspace="ApprovePurchaseOrderLines">
      <Description template="^1 ^2 - ^3 ^4 ^5" arguments="PurchaseOrderHeader.PurchaseOrderNumber PurchaseOrderHeader.Name1 PurchaseOrderLine.LineText PurchaseOrderLine.PriceCurrency  PurchaseOrderHeader.Currency"/>

      <Waypoint pane="PurchaseOrderLinesFilter">
        <Restriction title="Purchase Order Lines for Approval">
          <Match queryField="PurchaseOrderLine.InstanceKey" field="InstanceKey"/>
        </Restriction>
      </Waypoint>

      <Target pane="PurchaseOrdersCard">
        <Restriction title="Approvals">
          <Match queryField="PurchaseOrderLine.InstanceKey" field="InstanceKey"/>
        </Restriction>
      </Target>
    </Link>
  </Notifications>
</MNSL>
```

2. Change the notification setup in Maconomy. In the Notifications workspace, select the **Synchronous Notification** check box.

3. Specify arguments, waypoint keys, and target keys in the field reference section (the green portions in the mnsl-specification).

```
<?xml version="1.0" encoding="UTF-8"?>
<MNSL xmlns="http://www.deltek.com/ns/mnsl" version="0.2">
  <Notifications name="ApprovePurchaseOrderLine" title="Approve Purchase Order Line">

    <Link workspace="ApprovePurchaseOrderLines">
      <Description template="^1 ^2 - ^3 ^4 ^5" arguments="PurchaseOrderHeader.PurchaseOrderNumber PurchaseOrderHeader.Name1 PurchaseOrderLine.LineText PurchaseOrderLine.PriceCurrency  PurchaseOrderHeader.Currency"/>

      <Waypoint pane="PurchaseOrderLinesFilter">
        <Restriction title="Purchase Order Lines for Approval">
          <Match queryField="PurchaseOrderLine.InstanceKey" field="InstanceKey"/>
        </Restriction>
      </Waypoint>

      <Target pane="PurchaseOrdersCard">
        <Restriction title="Approvals">
          <Match queryField="PurchaseOrderLine.InstanceKey" field="InstanceKey"/>
        </Restriction>
      </Target>
    </Link>
  </Notifications>
</MNSL>
```

4. Go to Approval Hierarchies > Approval Hierarchy Rules.
5. Link the approval task to the notification type.

Security Service File Check

In Maconomy, it is possible to check if a file to be uploaded or downloaded can be accepted. One or more of the following predefined file check functions can be enabled.

Note: File check using Simple Magic is no longer supported. This produces an error in the Coupling Service log if this capability was previously enabled.

File Check Using an External Command

You can add one or more external commands to check a file for security. It is not mandatory to define either a allowlist or a blocklist.

Note that in the instructions below, the first command must have the value 1, and the following commands must be consecutive. If a number is missing, then the following commands are skipped.

To file check using an external command:

1. Open the **server.ini** file and add the following line:

   ```
   security.filecheck.external.<n>.command = <command with "{}" for file>
   ```

2. To define which exit codes are accepted, add the following line (default value is 0):

   ```
   security.filecheck.external.<n>.exitcodes = <comma-separated list of exit codes>
   ```

3. To define an allowlist, add the following line:

   ```
   security.filecheck.external.<n>.allowlist = <path to the allowlist file>
   ```

   Note: The allowlist contains the output that is accepted. Each line in the file contains an output text. If the output from the command do not contain one of these texts, then the file is not accepted. Default is not to use an allowlist.

4. To define a blocklist, add the following line:

   ```
   security.filecheck.external.<n>.blocklist = <path to the blocklist file>
   ```

   Note: The blocklist contains the output that is not accepted. Each line in the file contains an output text. If the output from the command contains one of these texts, then the file is not accepted. Default is not to use a blocklist.

5. Review the files. When creating one of the files it can be helpful to temporarily add this to the logback.xml file in the CouplingService\configuration folder:

   ```
   <logger
   name="com.maconomy.coupling.service.security.McFileCheckUsingExternalCommand" additivity="false">
       <level value="DEBUG" />
       <appender-ref ref="FILE" />
   </logger>
   ```

   and then try to upload files to be accepted (allowlist) or not accepted (blocklist). After, check the maconomy.log file in the CouplingService\log\coupling folder and update the relevant file.

Examples using external commands:

```
# Checks exit code == 0 and then white/black list
```

```
security.filecheck.external.1.command = C:\cygwin64\bin\file.exe -b "{}"
security.filecheck.external.1.allowlist = ./security-filecheck-external-allowlist-default.txt
security.filecheck.external.1.blocklist = <file2>

# Checks exit code == 0
security.filecheck.external.2.command = /path/to/anti/virus/checker "{}" --check

# Checks exit code == 42 or 43
security.filecheck.external.3.command = /strange/tool "{}"
security.filecheck.external.3.exitcodes = 42,43

# This one will be skipped because the previous command used n=3:
security.filecheck.external.5.command = /will/not/run "{}"
```

SYSTEM MAINTENANCE and REGULAR USE

Scheduled Background Tasks

See the Deltek Maconomy Background Task Guide for details.

Increase Allowed Number of Named Users

If you need to increase (or decrease) the number of licensed Maconomy users, you can now make the update yourself from within the Workspace Client, as opposed to having this performed as a chargeable activity by one of Deltek's technical consultants, requiring a restart of the system (which is no longer needed).

Contact your account manager so they can process your requested user extension order. Once that process is complete, they will provide you with a license key which is required for the following procedure.

Note that this procedure can only be performed by one of your system administrators, or any user who has access to the System Setup workspace.

To update your company's allowed number of named users:

1. Go to **Setup » System Setup workspace » Installation Details section » Installation Information tab » Version island**.
2. In the **Named Users Code** field, enter the license key provided by your account manager.

   Sample code:

   ```
   150--a32c675c-f5d7b068-0741a962-93367c4c
   ```

3. Click **Save**, or press ENTER.

   Maconomy immediately applies the update.

User Masking

The user, employee, and company masking feature enables you to mask personal information that is related to former employees. This feature deletes some relevant records according to your specifications and masks relevant fields in other records.

User names and employee names are stored in multiple places in Maconomy, such as contact person information, vendor information, user-specific selection criteria, setup and transaction data, documents, and various logs.

When you enable masking, only the user and/or employee information is masked, even when you use the company masking functionality. When you use the company masking functionality, the user and employee information that is associated with the specified company is masked.

This feature performs the following activities to mask personal information:

- Changes the user name to the value that you specify as the masking value. Updates all occurrences of the user name to the masked value. Deletes the user if it exists.
- Changes the name of the corresponding employee (including revisions), contact person, and vendor to the masked name and updates all occurrences of the original employee name to the masked name. Retains the employee record, but with all personal information masked.
- Clears all of the personal information for the employee, contact person, vendor, and many other relevant fields.
- Clears any personal information in logs of field changes to the user, employee, employee revisions, contact person, and vendor who are associated with the user.

> **Note:** The masking feature does not affect documents that may have been attached in Maconomy or other data that could hold employee names.
>
> For example, if you set up a G/L dimension named Entity that contains account manager information, and you need to mask the personal information for a former employee who is listed as an account manager (in the Entity dimension), and you have enabled the logging of changes to the Entity dimension, the personal information in the Entity dimension for the former employee **is not masked**. In addition, when you make a change in the Entity dimension, the original employee name appears in the log of changes.

If you have the correct access privileges you can:

- Mask information for one or more users and/or employees.
- Mask all user and employee information for one or more companies.

Masking-related workspaces and options are not displayed to users who do not have the correct access privileges.

User Masking Workflow

The high-level workflow for using masking is as follows:

1. In the System Setup workspace, set the value of the **Allow User Masking for Company** system parameter to enable masking.
2. In the Masking workspace, on the **User Masking** tab, enter the user and employee masks to be used and select the user names and/or employee numbers whose information is to be masked. See *Mask Users*.

3. In the Masking workspace, on the **Company Masking** tab, enter the user and employee masks to be used and select the company whose user and/or employee information is to be masked. See *Mask Companies*.
4. Run the masking routine via the appropriate action button.

   When the masking routine finishes, it displays a log file that contains a list of all of the major instructions that were executed. You can save the log file to a location that you choose. This is helpful if you need to retain proof that you performed the masking.
5. Examine the log file to check for any errors.

   You can remove these log files manually if no problems occurred.

> **Tip:** If a log file indicates that there were problems in the masking operations, send the log file to Deltek Customer Care.

User Masking Procedures

Mask Users

To perform user/employee masking, ensure that the following conditions are met:

- At least one of the columns User Name and Employee No. must contain a value.
- The values in those columns must be unique.
- The current user cannot be included in the list of users to mask.
- The user named Administrator cannot be included in the list of users to mask.

To mask a user name, complete the following steps:

1. Navigate to **Human Resources » Masking » User Masking** tab.
2. In the **User mask** field in the **Set default values** island, enter the string to use as the mask.

   The masking routine replaces all selected user names with this string. You can enter any string, with a maximum length of 50 characters.
3. In the **Employee mask** field in the **Set default values** island, enter the string to use as a mask.

   The masking routine replaces all employee names that have the selected numbers with this string. You can enter any string, with a maximum length of 50 characters.
4. Enter or search for a user name in the **User Name** column in the **User Masking** sub-tab.

   You can enter any string; the value in this field is not validated. This enables you to locate and mask information for user names that have already been removed from Maconomy, but that still exist in various locations, such as change logs and database tables.
5. Enter or search for an employee number in the **Employee No.** field in the **User Masking** sub-tab.

   The value in this field is validated; the employee number must exist in Maconomy.

> **Note:** You can enter just a user name, just an employee number, or both user name and employee number.

6. Click the **Perform User and Employee Masking** action button.

   A warning dialog box prompts you to confirm that you want to run the masking routine. If you do not click OK or Cancel, the action is automatically canceled.

> **Warning:** Masking actions are not reversible. Be sure that you want to run the masking routine with the values that you have specified before you click OK.

7. Click **OK** in the warning dialog box to run the masking routine.

> **Warning:** After you click OK, **do not interrupt the execution** of the masking routine.

The masking routine runs; this can take up to an hour or longer to complete, depending on the speed and the size of your database. A progress bar indicates how processing is going.

When the masking routine is finished, it displays a log file. You can save this log file.

If the masking routine encountered any errors, a message warns you that there are exceptions in the log file. If you receive this message, send the log file to Deltek Customer Care.

Mask Companies

To perform user/employee masking for a selected company (or companies), ensure that the following conditions are met:

- The **Company No.** column must contain a valid company number that exists in Maconomy.
- The value in the **Company No.** column must be unique.
- You can perform masking only for companies for which the **Allow User Masking for Company** system parameter is selected.
- The current user cannot be included in the list of users to mask.
- The user named Administrator cannot be included in the list of users to mask.

To mask user and/or employee information that is associated with the specified company or companies, complete the following steps:

1. Navigate to **Human Resources » Masking » Company Masking**.
2. In the **User mask** field in the **Set default values** island, enter the string to use as the mask.

   The masking routine replaces all selected user names with this string. You can enter any string, with a maximum length of 50 characters.
3. In the **Employee mask** field in the **Set default values** island, enter the string to use as a mask.

   The masking routine replaces all employee names that have the selected numbers with this string. You can enter any string, with a maximum length of 50 characters.
4. Enter or search for a company number in the **Company No.** field in the **Company Masking** sub-tab.

   The company number must exist in Maconomy, and masking must be enabled for this company via the **Allow User Masking for Company** system parameter.
5. Click the **Perform User and Employee Masking** action.

   A warning dialog box prompts you to confirm that you want to run the masking routine. If you do not click **OK** or **Cancel**, the action is automatically canceled.

> **Warning:** Masking actions are not reversible. Be sure that you want to run the masking routine with the values that you have specified before you click OK.

6. Click **OK** in the warning dialog box to run the masking routine.

> **Warning:** After you click OK, **do not interrupt the execution** of the masking routine.

The masking routine runs; this can take up to an hour or longer to complete, depending on the speed and the size of your database. A progress bar indicates how processing is going.

When the masking routine is finished, it displays a log file. You can save this log file.

If the masking routine encountered any errors, a message warns you that there are exceptions in the log file. If you receive this message, send the log file to Deltek Customer Care.

General Server 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.

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.

Running an Index Script

The database index handling process includes support for index scripts. While this can be used for a multitude of purposes, the main benefits are validating, creating, migrating, and maintaining indices in the Maconomy database.

An index script is a list of commands. The script language supports a single kind of variable: a list of index descriptions. These index-lists can be read from files, from the Maconomy database, or from the DbDesc and index-lists written to files or dumped as SQL.

Index script format:

- All commands are case-insensitive.
- All index-list variables are case sensitive.
- Two slashes are used for comments.
- Double quotes are used for filenames and print text.

To run an index script:

1. Create and read all files from \Maconomy\tmp.
2. Run an index-script as a MaconomyServer command line command, in the following format:

   ```
   MaconomyServer.exe -iMaconomyServer.<application> -S<shortname>
   --IndexScriptFile "\Tmp\IndexScrpt.txt"
   ```

   Example:

   ```
   MaconomyServer.exe -iMaconomyServer.w_21_0.sp102 -Smarco
   --IndexScriptFile "\Tmp\IndexScrpt.txt"
   ```

Note: If you are not a power user, you will likely find what you need in the examples at the end of this section. Skip the description of the 'Script commands' and go straight to the example section.

Index Script Parameters

Occasionally, it is more practical to use parameters for index scripts. You can specify these parameters at the end of the command line, and these will replace file parameter placeholders in the script. File parameter placeholders follow the following format: ${1} for the first parameter, ${2} for the second, and so on.

Example:

If you specify the following parameter:

```
MaconomyServer.exe -iMaconomyServer.<application> -S<shortname>
   --IndexScriptFile "\Tmp\IndexScrpt.txt" OldDB.json ListOld.txt
```

it replaces the following script:

```
ReadIndexFromJson jsonIndex ${1}
WriteIndexToText jsonIndex ${2}
```

When you run the index script, the IndexHandler reads the json-file OldDB.json and converts its contents into a more readable format before copying everything into the ListOld.txt file.

Script Commands

This section contains a list of the current index script commands.

Print

Print simply outputs the content of a quoted string to standard output.

Index-list variables cannot be used with print-only text

Use Print to track progress.

Example:

```
Print "Script completed"   // Outputs status text to the console
```

ReadIndexFromDatabase

This command accesses the shortname and builds an index-list containing all indices in the database.

Note this list also contains uniqueness constraints, since these are in fact also indices, but annotated in the index-list as such. That information can later be used to generate correct DDL for creating and removing such index/constraints.

Example:

```
// Read all index from database
ReadIndexFromDatabase dbIndex   // Save in an index-list named 'dbIndex'
```

ReadIndexFromDbDesc

This command works as the command above, but instead of reading indices from the database, it reads from the DbDesc. This contains (almost) all indices on the system, but not uniqueness constraints or instancekey constraints.

Example:

```
Print "Read all index from DbDesc "
ReadIndexFromDbDesc dbDescIndex // Save in an index-list named 'dbDescIndex'
```

FindIndexFromCreConstraint

This command generate an index-list of all uniqueness constraints. Since there is nowhere to read this information, the list is generated with an algorithm similar to what the server uses to generate uniqueness constraints using information from the DbDesc.

Example:

```
print "Generate creconstraint index"    // Generate uniqueness constraints
FindIndexFromCreConstraint tmp           // And save in an index-list named 'tmp'
```

Note that there is a server setting in Maconomy.ini / MaconomyCustom.ini to guide how deferred constraints are handled on SQL Server that lacks support for that.

- DeferredConstraintsOnSqlServer=Ignore *This setting disregards these indices and does not read them (default behavior if not specified)*
- DeferredConstraintsOnSqlServer=Unique *This setting reads the index as a normal unique constraint*
- DeferredConstraintsOnSqlServer= Error *This setting work as 'Ignore'. Deferred constraints are normal in the DbDesc and do not generate an error*

FindIndexFromInstanceKeyConstraint

This command generate an index-list of all instance key constraints. Since there is nowhere to read this information, the list is generated with a similar algorithm to that which the server is using to generate instance key constraints using information from the DbDesc.

Example:

```
print "Generate Instance Key Constraint"  // Similar to the way the server do it.
FindIndexFromInstanceKeyConstraint tmp2   // And save in an index-list named 'tmp2'
```

ReadIndexFromDbDescAll

This command combines the three previous commands: ReadIndexFromDbDesc, FindIndexFromCreConstraint, and FindIndexFromInstanceKeyConstraint. It generates the three index-lists and combines them into a single list that is the sum of the three.

Example:

```
print "Build index-list from DbDesc-index and constraint"
ReadIndexFromDbDescAll dbDescIndex  // And save in an index-list named ' dbDescIndex'
```

Note: See the description of DeferredConstraintsOnSqlServer setting in the FindIndexFromCreConstraint section.

WriteIndexToText

The Json file format used in the commands above are human-readable, but imperfect for a condensed overview of the involved index. An alternative is to write out an index-list variable to a text file. This format is preferable for a quick overview of an index-list and the attributes of the involved index.

Example:

```
Print "Save TestJason as a text file for easy reading"

// Take the index-list 'TestJason' and save as a text file
// to \Maconomy\tmp\TestJason.txt

WriteIndexToText TestJason TestJason.txt
```

This is an example of the generated output:

```
Relation DEV_PROGRESSSNAPSHOT has the following index :
  Index IKDEV_PROGRESSSNAPSHOT, Non-clustered, Unique, Table constraint: INSTANCEKEY -
Asc;
Relation MACONOMY_USERS has the following index :
```

```
  Index MACONOMYUSERSINDEX2, Non-clustered: SESSIONID - Asc; ROLEINSTANCEKEY - Asc;
```

WriteIndexToJson

This saves an index-list to a Json file. The format of this file is not a part of this document. This is a method of saving an index list for later reference.

Example:

```
Print "Save found duplicates as a json file"

// Take the index-list 'duplicates' and save as a
// .json file to \Maconomy\tmp\FoundDups.json

WriteIndexToJson duplicates FoundDups.json
```

This is an example of a start of an index-list Json-file:

```
{
  "Version" : "1.2",
  "Relations" :
  [
    {
      "Relation" : "ABSENCEBOOKINGDETAIL",
      "Index" : "ABSENCEBOOKINGDE02",
      "Type" : "Non-clustered",
      "Unique" : "No",
      "TableConstraint" : "No",
      "InitiallyDeferred" : "No",
      "Columns" :
      [
        {
          "Name": "EMPLOYEENUMBER",
          "Sorting": "ASC",
          "Included": "No"
        },
        {
          "Name": "ABSENCETYPE",
          "Sorting": "ASC",
          "Included": "No"
        },
        {
          "Name": "PERIODSTART",
          "Sorting": "ASC",
          "Included": "No"
        }
      ]
    },
    {    {
      "Relation" : "ACTIVITY",
      "Index" : "UQACTIVITY",
      "Type" : "Non-clustered",
      "Unique" : "Yes",
      "TableConstraint" : "Yes",
      "ConstraintImplementationIndex" : "ACTIVITY01",
      "InitiallyDeferred" : "No",
      "Columns" :
      [
        {
          "Name": "ACTIVITYNUMBER",
```

```
          "Sorting": "ASC",
          "Included": "No"
        }
      ]
    },
```

ReadIndexFromJson

This is the reverse of the above command: It reads a Json file with an index-list definition and saves it to an index-list structure.

Example:

```
Print "Read index from a json file"

// Read an index structure from a Jason-file named "TestJason.json" in \Maconomy\tmp
// And then store it in an index-list named 'jsonIndex'

ReadIndexFromJson jsonIndex TestJason.json
```

Note: There is a server setting in Maconomy.ini / MaconomyCustom.ini describing how deferred constraints are handled on SQL Server that lacks support for that.

- DeferredConstraintsOnSqlServer=Ignore *This setting disregards these index commands and does not read them (default behavior if not specified)*
- DeferredConstraintsOnSqlServer=Unique *This setting reads the index as a normal unique constraint*
- DeferredConstraintsOnSqlServer= Error *This setting creates an error if a deferred constraint is read from a json-file*

AddIndex

Use this to add to add two index-lists together.

Note: It is possible that both index-lists contain an index with the same name and possibly different definitions. In that case, the index from the first list is overwritten with the index definition from the second. For example, in the example below, if both index-list contain an index on ACCOUNT named TEST, the definition from the index-list named 'Tmp' is used.

Example:

```
Print "And add all the index in Tmp to those in dbDescIndex"

// This function adds two index-lists together (based on name only)
// and saves the result in the first (dbDescIndex)

AddIndex dbDescIndex Tmp
```

FilterIndex

Use this command to run through an index-list and extract all indices to a new list with that quality.

This is the complete list of qualities to filter on:

- Clustered
- Notclustered
- FunctionBased
- NotFunctionBased

- Unique
- NotUnique
- Constraint
- NotConstraint
- Deferred
- NotDeferred
- Standard
- NotStandard

Example:

```
Print "Filter out in sub-sets based on type"


// Read all index from database

ReadIndexFromDatabase dbIndex   // Save in an index-list named 'dbIndex'


// This function adds two index-lists together (based on name only)
// and saves the result in the first (dbDescIndex)

FilterIndex dbIndex UniqueIndex Unique           // UniqueIndex: a list of unique index in DB
FilterIndex dbIndex ClusteredIndex Clustered     // ClusteredIndex: a list of clustered index
FilterIndex dbIndex FunctionBasedIndex FunctionBased   // FunctionBasedIndex. a list of FB Index
FilterIndex dbIndex NonDefConstraint NotDeferred // NonDefConstraint: a list of constraints
                                                 // that are not initially deferred
```

CopyIndex

Use this command to make a copy of an index-list.

Example:

```
Print "Assign the index-list dbDescIndex to the new index-list variable AlsoDbIndex "


CopyIndex AlsoDbIndex dbDescIndex
```

RemoveIndexFromList

This command deletes all indices named in the index-list given in the second argument from the index-list in the first argument. Therefore, this is a kind of subtraction function.

Note:this removal is based on the index name only,not on the detailed definition of the index (columns, and so on).

Example:

```
Print "Remove index used in hint from the 'Duplicates' list"
Print "Later we don't want to drop those"

RemoveIndexFromList Duplicates IndexFromHints
```

CompareIndex

This command compares two index-lists and as a result create three new index-lists. One includes the index that was added and one includes a list of the indices that were missing, and both are from the

perspective of the first argument index-list. Finally, an index-list containing all indices that were present in both the argument index-lists, but differ in definition (for example, both lists have an index called TestIndex, but spanning different sets of columns. Alternatively, only one may have a unique index).

Example:

```
print "Compare DbDesc to database"

// Compare two index-lists and create 3 new based on the result.
// These contains the added, missing and changed index.
// From the perspective of the first index-list.

CompareIndex Database InstallReference addedIndex missingIndex changedIndex
```

MarkAsStandard

Some indices are considered standard. These are preferable to non-standard indices if you are looking for duplicates to drop (see FindIndexDuplicates), everything else being equal between the indices under consideration.

This command can mark all indices in an index-list as standard.

(Standard status can be applied to jso-files and read from json-files.)

Run the command as follows:

```
Print "Mark index read from json as standard"

MarkAsStandard standardIndexNew  // All standard index is marked as such
```

When you eliminate duplicates (see FindIndexDuplicates), standard indices will not be chosen as a duplicate (everything else being equal between the indices under consideration).

Example:

```
Print "Now mark all as standard"

MarkAsStandard jsonIndex
```

FindIndexDuplicates

Many databases contain duplicate index definitions, which means that the same selection, sorting, and order of columns are contained in several indices. This provides no benefit and has a performance impact when updating the table, and should therefore be avoided. This command can identify duplicate index in an index-list, and decides on the index to keep, using the following rules:

- Keep clustered over non-clustered
- Keep unique index over non-unique
- Keep table constraints over indices
- Keep a superset of columns (keep an index on column a, b and c over an index on a and b)
- Keep “normal” columns over included columns (SQL Server)
- As a tie braker: Keep index marked as ‘Standard’ over the non-standard index. See MarkAsStandard.
- As a tiebreaker: If one index is a standard index and the other one is not, keep the standard index

Example:

```
Print "Finding all duplicates in the index-list DatabaseList"

// Find all duplicates in the index-list 'DatabaseList'
// and save the findings to a new structure called 'duplicates'
```

```
FindIndexDuplicates DatabaseList duplicates
```

RenameInstanceKeyConstraints

From Maconomy 2.4 to 2.5, the naming scheme for instance key constraints changed.

- In 2.4 and older versions, an instance key constraint would have the name INSTANCEKEYnnn where nnn is a 3-digit internal number of the underlying table.

  Example: INSTANCEKEY408 is the instance key constraint for the ACTIVITYREFERENCE table.

- In 2.5 and later versions, an instance key constraint would have the name 'IK' followed by the name of the underlying table.

  Example: IKACTIVITYREFERENCE is the instance key constraint for the ACTIVITYREFERENCE table.

The RenameInstanceKeyConstraints command will take a list of indices and constraints, and rename all old-style named instance key constraints according to the new scheme. At the same time, it will create an SQL script for updating the names of the instance key constraints in the database according to the new naming scheme. The script will match the database in use.

Example:

```
print "Change all instance key constraints to new naming"

// Renames old-style instance key constraints if any
RenameInstanceKeyConstraints OriginalDatabaseIndex "renameInstanceKeyConstraints.sql"
```

WriteIndexAsDropStatements

In the end, you will likely need to convert index-lists to SQL and the following commands are examples of that. WriteIndexAsDropStatements can generate a file of drop statements based on an index-list (for example, a list of duplicates generated by the FindIndexDuplicates command). All files are generated in the \maconomy\tmp folder.

Example:

```
print "Write all duplicates as DROP statements"

// Take the index-list 'duplicates' and create DROP statements
// and write them to \Maconomy\tmp\DropDuplicates.sql

WriteIndexAsDropStatements duplicates "DropDuplicates.sql" MACINDEX
```

That last parameter is a tablespace parameter and is only used for Oracle. For Oracle, we might need to drop and recreate constraints using the argument index to be dropped. When creating the constraints again we need a tablespace. An argument of 'default' creates the constraint without a tablespace parameter and it is created in the default tablespace.

WriteIndexAsDisableStatements

This command creates a SQL file in \Maconomy\tmp, which disables all indices in the argument index-list.

Example:

```
print "Write all duplicates as Disable statements"

// Take the index-list 'duplicates' and create DROP statements
// and write them to \Maconomy\tmp\DropDuplicates.sql

WriteIndexAsDisableStatements duplicates "DisableDuplicates.sql"
```

WriteIndexAsCreateStatements

This command creates a SQL file in \Maconomy\tmp with CREATE-statements for all indices in the argument index-list. You need to specify a table-space (Oracle) or filegroup (SQL Server) for the create statements to use OR use the keyword DEFAULT to specify that no tablespace should be included in the CREATE-statement.

Example:

```
print "Write all index in the index-list 'Tmp' as CREATE statements"

// Create all index in the default tablespace
WriteIndexAsCreateStatements Tmp "CreateIndex.sql" DEFAULT
```

WriteNormalizeConstraintsAsSql

On Oracle, unlike SQL Server, a constraint will not create its own index to implement the constraint, but instead depends on an existing index if at all possible (even a non-unique one). This command create a script that follows this process:

1. Drops all constraints depending on “external” index
2. Drops the constraints
3. Creates all the above constraints again
4. Create all the above indices again

In extreme cases, it might be necessary to run the process (create the script,un the script) multiple times.

The last parameter is the tablespace for the index / constraints to be created in or the word ‘default’ for them to be created in the default tablespace.

This command creates an empty script file on SQL Server.

Example:

```
print "Create a script that makes no Oracle constraint depend on an ‘external’ index"

// Create all index in the default tablespace
WriteNormalizeConstraintsAsSqlTmp "Normalize.sql" DEFAULT
```

ParseExpandScript

When running expand scripts tables make change name or be rebuilt. Previously, the standard procedure was to drop all indices, upgrade by running expand scripts, and then rebuild all indices. However, rebuilding all indices is time-consumingNow we script an alternative: Parsing an expand script to find all the indices that must be dropped and create a script for that purpose. All other indices can be kept, and later in the process, we can recreate the dropped index.

Example:

```
Print "Prepare for upgrade"

// Read the Expand script expand.w200sv0-w210sv2.sql
// and create an index-list for indexes to be dropped now
// and also an index-list for index to be created later
ParseExpandScript dbDatabase expand.w200sv0-w210sv2.sql toBeDroppedNow
toBeCreatedLater
```

FindAllHintIndex

When planning to drop index (for example, a duplicate index), it is important to determine if it is used in a hint. This command searches through SQL Server hint definitions (Oracle is not supported) and build an index-list of indexes used in hints. Since a index mentioned in a hint does not contain any details about the index, the FindAllHintIndex command needs an argument index- list that defines the hints—typically created by reading the index from the database using ReadIndexFromDatabase. An error is reported if an index is mentioned in a hint but not defined in the argument index structure.

Example:

```
Print "Find all index in Hints files"

// First build an index-list of all index in the database
ReadIndexFromDatabase dbDatabase // Save in an index-list named 'dbDatabase'


// Then read the hints files and find all index there.
// If an index is not defined in the first argument: report error
// Store the found indexes from hints in 'IndexFromHints'
FindAllHintIndex dbDatabase IndexFromHints
```

FindRenamedIndex

This command, which takes an index list and compares it with another, is uncommon but could be needed. It. The command finds an index from the first list that cannot be found in the second, but does exist with a complete identical definition, but with another name, are considered to be renamed. The command then generates a SQL script for renaming these files.

Example:

```
Print "Find renamed index in database"

// First build an index-list of all index in the database
ReadIndexFromDatabase dbDatabase // Save in an index-list named 'dbDatabase'

// Compare index in 'dbDatabase' with index in 'newVersion'
// Identify renamed index and store them in a new index-list 'renamed'
// Also create a SQL script for renaming the index in 'dbDatabase': 'Renamed.sql'
FindRenamedIndex dbDatabase newVersion renamed Renamed.sql
```

ValidateConstraints Command Line Option

You do not need to create an index script to validate that all the vital uniqueness constraints are present. All you need to do is run a server as follows:

```
\Maconomy\bin\MaconomyServer.<Application>.cmd -S<shortname> --ValidateConstraints
```

Example:

```
\Maconomy\bin\MaconomyServer.w_22_0.sp100.cmd -Sm22p0 --ValidateConstraints
```

If everything is fine, the command will return 0 to the environment and display the following report:

```
Constraint check successful. No constraints are missing.
```

If constraints are missing, the command will display the following report:

```
Constraint check failed.
```

```
Create missing constraints by running <file>
```

Example:

```
Constraint check failed.

Create missing constraints by running
C:\maconomy\Tmp/CreateMissingConstraints.m22p0.sql.
```

If the missing constrains must be created in a specific Oracle tablespace or SQL Server file group, you can specify this on the command line as follows:

```
\Maconomy\bin\MaconomyServer.w_22_0.sp100.cmd -Sm22p0 –ValidateConstraints MacIndex
```

where 'MacIndex' is an example of a tablespace or file group name.

Script Examples

Example 1: Eliminate Duplicate Index

Use the script below to eliminate indices in the database, but keep those used in hints.

```
// Eliminate Duplicates in database

print "Read index from database" // Print status to stdout
readIndexFromDatabase dbDatabase // Read all index from database and save in a index-
structure and name it 'dbDatabase'

print "Write all index in text format to  dbDatabase.txt"
WriteIndexToText dbDatabase dbDatabase.txt

print "finding duplicates"
FindIndexDuplicates dbDatabase duplicates      // Find all duplicates in the index-
structure 'dbIndex1' and save the finding to a new structure called duplicates

print "Drop duplicates as text"
WriteIndexToText duplicates "Duplicates.txt"   // Take the index-structure
'duplicates' and create DROP statements and write them to
\Maconomy\tmp\DropDuplicates.sql

print "Find all index in Hints files"
FindAllHintIndex dbDatabase IndexFromHints     // Read the hints file and find all
index there. If not in the DB report error

print "Write list of index index in hints files to \Maconomy\tmp\IndexFromHints.txt"
WriteIndexToText IndexFromHints IndexFromHints.txt

print "Remove index used in hint from the duplicates list: we don't want to drop
those"
RemoveIndexFromList duplicates IndexFromHints

print "Write the final list of index to drop to \Maconomy\tmp\DuplicatesWoHints.txt"
WriteIndexToText duplicates DuplicatesWoHints.txt

print "Also dump the final list of index to drop as Json to
\Maconomy\tmp\DuplicatesWoHints.json"
WriteIndexToJson duplicates DuplicatesWoHints.json

print "And finally create drop-statements for  index to drop to
\Maconomy\tmp\DropDuplicatesWoHints.Sql"
WriteIndexAsDropStatements duplicates DropDuplicatesWoHints.Sql
```

Review and run the script '`DropDuplicatesWoHints.Sql`' to remove all duplicate index NOT used in a SQL Server hints file.

Example 2: Create and Use an Index Reference File

After upgrading and tuning a system, a consultant takes a snapshot of the current index.

```
// Take a snaphot

print "Read index from database"
// Read index from database and save in an index-structure and name it 'dbIndex1'
readIndexFromDatabase dbIndex

print "Save it as a json file"
WriteIndexToJson dbIndex "dbIndex-june-2020.json"

print "Also save found index as a text file for easy reading"
WriteIndexToText dbIndex dbIndex-june-2020.txt
```

Four months later he comes back, because they have a number of issue with the system.

He decides to check if the index are, as he left them.

```
// Take a new snaphot and compare with june index

print "Read index from database"
// Read index from database and save in an index-structure and name it 'dbIndex1'
readIndexFromDatabase dbIndex

print "Save it as a json file"
WriteIndexToJson dbIndex1 dbIndex-october-2020.json

print "Also save found index as a text file for easy reading"
WriteIndexToText dbIndex1 dbIndex-october-2020.txt

print "Read index from june database "
ReadIndexFromJson jsonIndexJune "dbIndex-june-2020.json"

print "Compare database index to june index"
CompareIndex dbIndex dbIndex2 addedIndex  missingIndex changedIndex

print "Save addedIndex as a text file for easy reading"
WriteIndexToText addedIndex addedIndex.txt     // Take the index-structure
'addedIndex' and save as a text fileto \Maconomy\tmp\addedIndex.txt

print "Save missingIndex as a text file for easy reading"
WriteIndexToText missingIndex missingIndex.txt // Take the index-structure
'missingIndex' and save as a text fileto \Maconomy\tmp\missingIndex.txt

print "Save changedIndex as a text file for easy reading"
WriteIndexToText changedIndex changedIndex.txt // Take the index-structure
'changedIndex' and save as a text fileto \Maconomy\tmp\changedIndex.txt

print "Save addedIndex also as a json file for easy reading and possibly diffing"
WriteIndexToJson addedIndex addedIndex2.json   // Take the index-structure
'addedIndex' and save as a text fileto \Maconomy\tmp\addedIndex.txt
```

Example 3: Handle Index During an Upgrade

The script below could be helpful as part of an upgrade.

```
// Start of upgrade

print "Read index from database" // Print status to stdout
readIndexFromDatabase dbDatabase // Read all index from database and save in a index-
structure and name it 'dbDatabase'

print "Write all index in text format to  dbDatabase.txt"
WriteIndexToText dbDatabase dbDatabase.txt

print "Prepare for upgrade"
ParseExpandScript dbDatabase expand.w200sv0-w210sv2.sql toBeDroppedNow
toBeCreatedLater    // Read the Expand script and create structures for index to be
dropped now and index to be created later

print "Drop some index pre upgrade using \Maconomy\tmp\toBeDroppedNow.sql"
WriteIndexAsDropStatements toBeDroppedNow "toBeDroppedNow.sql"     //

print "Write list of index to be dropped to \Maconomy\tmp\toBeDroppedNow.txt"
WriteIndexToText toBeDroppedNow toBeDroppedNow.txt

print "Create new index post upgrade using \Maconomy\tmp\toBeCreatedLater.sql"
WriteIndexAsDropStatements toBeCreatedLater "toBeCreatedLater.sql" // Take the index-
structure 'duplicates' and create DROP statements and write them to
\Maconomy\tmp\DropFoundDups.sql

print "Write list of index to be dropped to \Maconomy\tmp\toBeCreatedLater.txt"
WriteIndexToText toBeCreatedLater toBeCreatedLater.txt
```

After, run:

- toBeDroppedNow.sql
- expand.w200sv0-w210sv2.sql
- toBeCreatedLater.sql

Example 4: Compare Index from Database with Standard

Run this script to compare the index from the database with a standard index to…

The standard index for an APU is defined in this file:
MaconomyDir\Database\StandardIndexesAndConstraints.json

But for the below script to work, it has to be copied to \Maconomy\tmp

Then run this script:

```
// Compare database index to APU standard
// First copy file from MaconomyDir\Database\StandardIndexesAndConstraints.json
// to \Maconomy\Tmp

// Read the standard index for the installed APU
// from \Maconomy\Tmp\StandardIndexesAndConstraints.json
print "Read Std index from from APU"
ReadIndexFromJson apuIndex "StandardIndexesAndConstraints.json"

print "Read index from database" // Print status to stdout
readIndexFromDatabase dbDatabase // Save all index from Db in 'dbDatabase'

print "Compare APU to database"  // Comapare dbDatabase and apuIndex
CompareIndex apuIndex dbDatabase AddedIndex  MissingIndex ChangedIndex
```

```
print "Dump create script for missing index to \Maconomy\tmp\CreateMissingIndex.sql"
WriteIndexAsCreateStatements MissingIndex "CreateMissingIndex.sql" default

print "Also write missing to readable \Maconomy\tmp\MissingIndex.txt"
WriteIndexToText MissingIndex MissingIndex.txt

print "Dump drop script for added index to \Maconomy\tmp\DropAddedIndex.sql"
WriteIndexAsDropStatements AddedIndex "DropAddedIndex.sql"

print "Also write missing index to readable \Maconomy\tmp\AddedIndex.txt"
WriteIndexToText AddedIndex AddedIndex.txt

print "Finally write changed index to readable \Maconomy\tmp\ChangedIndex.txt"
print "Please inspect ChangedIndex.txt and consider if theese changes are desirable"
WriteIndexToText ChangedIndex ChangedIndex.txt
```

Review the changes from standard in addedIndex.txt. It always contains indices for the Maconomy_Users table and this is acceptable. If any additional indices are found and considered undesirable drop SQL can be found in DropAddedIndex.sql.

Review missing index listed in missingIndex.txt. Be especially aware of any removed constraints (starting with ‘IK’ or ‘UQ’). Consider creating missing index using SQL in CreateMissingIndex.sql.

Finally, review any changed index in the file changedIndex.txt. Consider: Why these changes, if any? Should standard be restored?

Checklists

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:

1. 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`.

2. 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.

    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.

Warning: 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
  ```

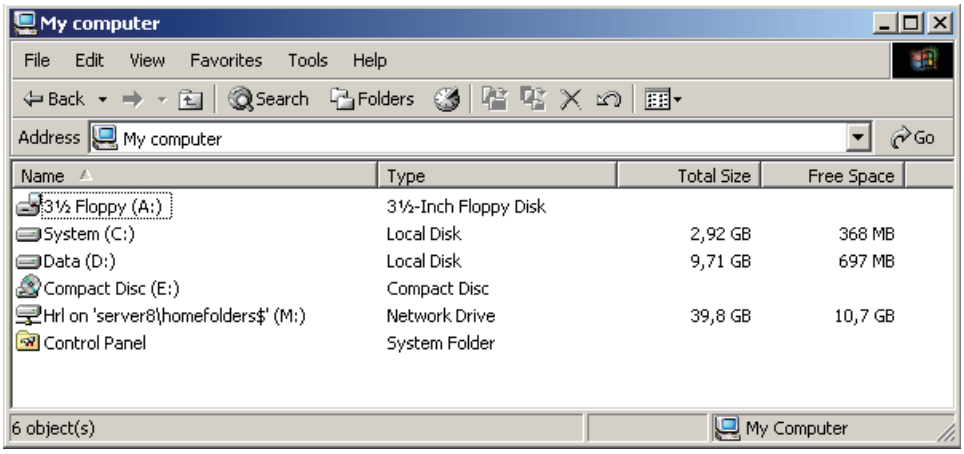
  The result may differ a bit depending on the Unix version, but it will show the results in 1 KB blocks.

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.”


My computer
File Edit View Favorites Tools Help
Back Search Folders
Address My computer
Go
Name Type Total Size Free Space
3½ Floppy (A:) 3½-Inch Floppy Disk
System (C:) Local Disk 2,92 GB 368 MB
Data (D:) Local Disk 9,71 GB 697 MB
Compact Disc (E:) Compact Disc
Hrl on 'server8\homefolders$' (M:) Network Drive 39,8 GB 10,7 GB
Control Panel System Folder
6 object(s)
My Computer

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.

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.

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.

Maintenance 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
    - Empty the target company
    - Export the data from the source company
    - Import the data into the target company
    - Verify that you can log in to the target company
- Make complete system restore
- Clean the tape station
- Replace backup tapes

Common Maconomy Server 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:

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.

Server Debug Output

Warning: Activating debug output can negatively affect server performance**.**

Debug output for the Maconomy Server can be activated globally with the ‘-d’ option in the server initialization file (MaconomyServer.<appl_ID>.I).

This activates a default set of test masks in the Maconomy server processes, and each process writes debug output to a process-specific log file in the ‘Tmp’ folder of the Maconomy installation.

The option only applies to server processes started after the option is set, so you must restart the Coupling Service to activate debug output for all server processes.

As of Maconomy 2.3 GA, you can also activate debug output from the Coupling Service for the already-running server processes owned by that Coupling Service.

Activate Server Debug Output

In this mode, the Maconomy server processes write debug output to user-specific log files in the ‘Log’ folder of the Maconomy installation.

To activate server debug output:

1. Open the ‘server.ini’ configuration file of the Coupling Service.
2. Locate the setting ‘server.debug.enabled’.
3. Remove the comment marker.
4. Set the setting to **true**.

```
server.debug.enabled = true
```

The debug output is immediately enabled for all subsequent server operations.

Reduce Performance Penalty of Debug Output

To reduce the number of log files and number of users affected by the performance penalty of debug output, collect only debug output for specific users.

The filter condition for matching individual users is similar to the ‘user-principal’ filter option in the MPM settings.

To filter debug output:

1. Open the ‘server.ini’ configuration file of the Coupling Service.
2. Locate the setting ‘server.debug.filters’.
3. Remove the comment marker from the embedded ‘principal-name’ filter setting.
4. Write a string expression that matches the users to collect debug output for:

```
server.debug.filters {
  principal-name = <string-expr>
}
```

Use this syntax of a string expression:

string-expr: [`contain(s)|match(es)`] [regex] <pattern>

Note: The logical combinators '`and`' '`or`', and '`not`' can also be used.

To disable debug output:

1. Open the 'server.ini' configuration file of the Coupling Service.
2. Locate the setting 'server.debug.enabled'.
3. Add a comment marker in front of the setting

   or

   Set the setting to **false**.

   ```
   server.debug.enabled = false
   ```

See Maconomy Server Frequently Asked Questions for more details.

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.

Warning: 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.”

Advanced Logging

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.

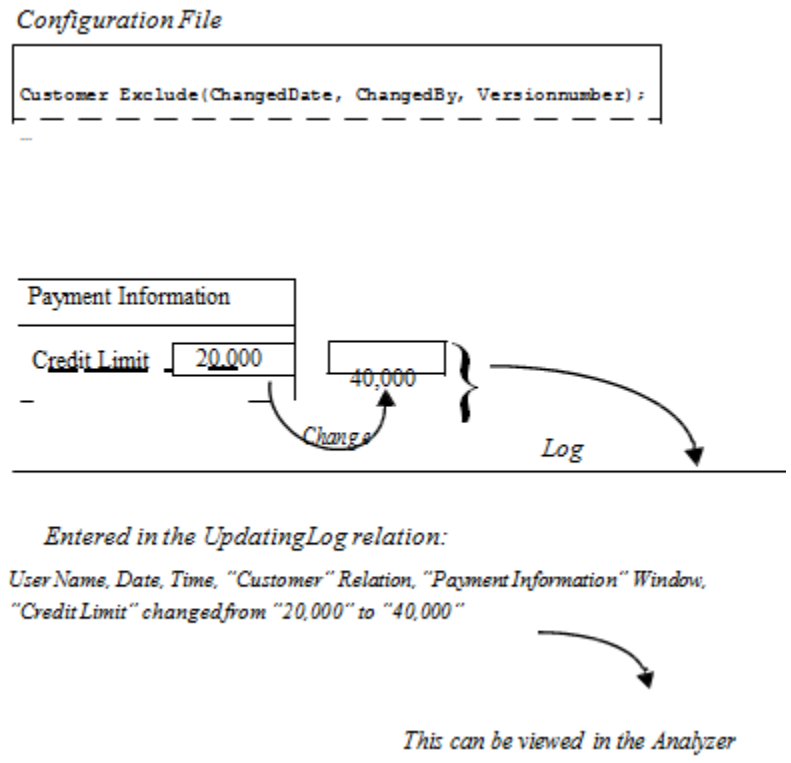
General Functionality

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.

MDoc / Data Dictionary

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.

   Note: 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.

   Note: 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.

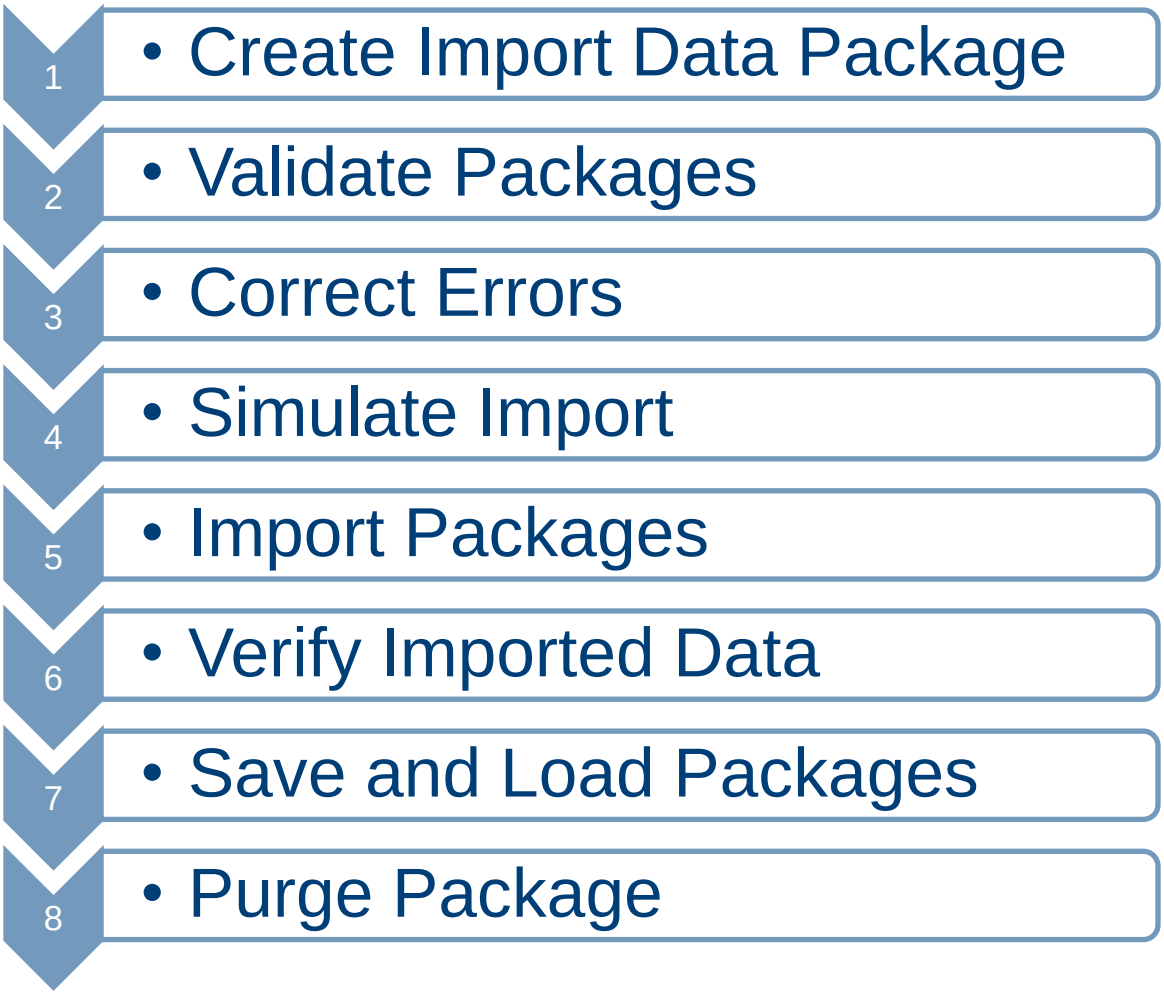
Data Import Packages

The Data Import Package process in Maconomy streamlines the import process. This is especially helpful at the beginning of a project, where a customer is new to Maconomy and wants to move large amounts of data from their old system to Maconomy (including employees, projects, historical information, and so on). Previously, importing was done through MConfig, or dialog by dialog. The new process introduces the concept of a Data Import Packages, which enables you to specify which files to include in a package, and transfer numerous import components at once. Additionally, you can validate the package prior to import to flag and correct any errors prior to import. Finally, you can save the packages in different formats for future use. A new Data Import Workspace provides a centralized place in the Workspace Client to create, validate, and import data packages.

This section includes:

Data Import Packages Workflow

This section describes the workflow for using data import packages and the related workspace.



Data Import Package Concepts

Data Import Package

Create Import Package Procedures

A Data Import Package is a collection of import files you put together in a specific order to create a single package to transfer data to Maconomy. This is particularly helpful as an efficient means to transfer legacy data to Maconomy. For example, a company may have subsidiary companies or branches all over the world. Data Import Packages provides a process to create initial packages and then replicate the packages to move data from all their legacy systems to the Maconomy system.

Additionally, a data import package contains metadata, such as checksums, which are a small size data derived from the import files which allow you to verify the integrity of the package and track it across different systems.

You can create data import packages in Workspace Client, then save and download the package from one Maconomy system and load it into another Maconomy system. As part of this process, master data as well as transactional data is created when you execute the import of the data import package. You can quickly create, update, or delete multiple records in different tables without navigating specific records. Moreover, the import files dictate how the data should be changed, acting as a trace of what has been imported into a given system. Additionally, the import files can be prepared by one user and delivered to another user who can then import them. Lastly, import files can be prepared once and then imported into multiple different systems allowing to you to maintain consistency between systems.

Use the Data Import Packages workspace to maintain the import data and automate the actual importing of the data files.

Color Coding

Colors of both text and highlight have meaning. Refer to the Legend in **Data Import Packages » Data sub-tab » Sliding Panel » Legend** for details.

Create Packages

Create Data Import Package Procedures

You can create import data packages for one-time use, or create them and use them in the future as template import packages, which is helpful if you are importing data from many different branches and will always include the sample import programs.

Creating a package requires these main steps:

- **Set up the package information.** Set up the package with a unique name for easy identification, and a short description that will be helpful to you and others viewing the package. Also, since import data is usually sensitive, you may also set up access control.

  Note: Both the Name and Description fields are editable.

- **Use the New Data Import Package action** to create the package. You must first create the package before you can attach the import files.

- **Complete the table lines** for the new package. Data import package lines represent import data that must be imported, as well as the order in which they must be imported. The table lines define this order.
- **Use the Attach File action** to select files to import and attach them to the new package. Create the package lines in the order in which you want to import them.

For example:

If you want to import new vendors to your Maconomy system, begin by specifying an Import Program. Import program refers to the dialog or container that manages the import data in the given import file. In this case, choose the Import Vendors container.

Next, add the files in the order you want to import them to account for data dependencies. A typical order may be customer, job, and so on. Finally, import the file after you have already imported vendors, since vendor invoices depend on vendors.

Option Lists

Two fields**, Option List** and **Selected Value**, support option lists in Data Import Packages and function as they do elsewhere in Maconomy.

1. **Option List**—Lists all the options that are defined in Maconomy and allows you to select one of them.
2. **Selected Values**—Once the option list is selected, then in the field **Selected Value** you can choose one of the values that are defined in the chosen option list. All the option lists and the corresponding values can be seen in the Option Lists dialog.

One difference when you use the fields in data import packages is that when you save a package to file, the specified option list and the selected value are also saved to that file. This means that when you load the saved package on another system, it is loaded with the same option list values.

If an option list does not exist on a system where the package is being loaded, these rules apply:

- If package is being loaded on another system, but the specified Option List does not exist on that system then both the Option List and Selected Value values are reset (set to empty value) before loading it.
- If Option List does exist, but Selected Value does not exist, then only Selected Value is reset before loading it.

In both cases, a warning is issued to users so that they can choose to continue with the load or stop.

Validate Packages

Validate Package Procedures

The Data Import Packages process allows you to validate packages so that you can find and resolve errors prior to import. Validating is much faster than importing, so in a short span of time you can troubleshoot any issues in preparation for the more time-consuming import. Validation can help to reveal obvious mistakes in the import file, such as using string value where integer/amount/real was expected, referencing non-existing popup, or referencing non-existing columns. Validate the files that you have attached to see if they are correct or if you have any errors.

Note that validation does NOT find business logic errors or foreign key error (such as specifying a negative billing price for an employee, using blocked activity, referencing non existing dimension, and so on).

To perform a more in-depth validation and find errors described above, use the Simulate Import action.

While you are validating, you can continue working with the lines that have not been marked for validation or create new package lines. The data import package lines that have been validated will have a validation log attached to them. To view the validation log, click the **View Validation Log** action either directly on the line or in the sliding panel.

If validation log contains errors, you can edit the import file by modifying the original file stored locally on your computer. Or, download the attached import file to your local computer, edit it, save it locally and then re-attach the corrected file to the package line and re-run the validation.

Overwriting Files

Note that the existing import file is overwritten with the new file, meaning that the original import file is permanently replaced. If you need revision control or history tracing, store your files in a source code repository, such as Git repository.

Hierarchy of Validation Example

You may wish to validate in a number of systems prior to the full import. We recommend a hierarchy of validation, moving to the next system when everything has passed at each stage, though the complexity of your environment will likely determine the levels of validation. For example, a small customer may have only a Test System and a Production System, whereas a larger customer may have a Test System, Pre-Production System, Development System, and Production System.

Best Practice: While it is not mandatory to perform validation before executing the import of a line, it is recommended as best practice. For example, otherwise you may discover that an import failed and needs to be repeated because on line 5671 there was a format error that could have been easily discovered by running a format validation.

If a more in depth validation of the package is required, use the **Simulate Import** action instead of or in addition to the validation.

Simulate Import

Simulate Import Procedures

Use the **Simulate Import** action instead of or in addition to the validation. This action performs an actual import of the file, however, it will not save any changes in the system when simulation completes. Simulating an import can help uncover errors that could not be found by simple format validation.

The **Simulate Import** action perform validation of only the current package line.

Note: This action does not update validation status or the validation log on the line. However, it does produce a validation log at the end, which is available for investigation.

Import Packages

Import Packages Procedures

Use the **Import** action to import selected lines on the data import package. It imports the lines in the order they are defined in the sub-tab by moving sequentially through the line list.

Note: It is not possible to make any changes to the lines that are marked for importing while the import is in progress.

If the import fails then the whole data import package execution terminates, and you must investigate the attached import log, which lists the reasons why the import failed.

For example:

Line 1 ← Completed

Line 2 ← Failed

Line 3

Since the import of Line 2 failed, the whole execution stops, so Line 3 is not picked up for import. This is because Line 3 might depend on Line 2, and if Line 2 failed then there is a high chance that Line 3 will also fail.

Similar to the validation flow, you can either edit the original file stored on your computer locally or download the currently-attached import file, correct it, re-attach it to the line, and restart the import. The import resumes from the next line that was not imported.

When a data import package line is successfully imported, you can no longer change it or any line above it. Instead of changing the imported package line, you can create a new one, listing all the changes you want to perform and then import that line.

Files import sequentially, in the order the package lines are defined in the sub-tab. If you are confident that you have independent import data files, you can create multiple data import packages and run them in parallel. This might be useful when importing data for different companies in the system. Blocking is less likely to occur when resources like system numbers are not shared.

Best Practice: we recommend that import users have a good understanding of Maconomy before trying to import multiple packages at the same time. While importing multiple packages at the same time utilizes parallelism and thus makes importing faster, if different packages block each other this efficiency will not be achieved.

Additionally, when importing a package, we recommend that you configure email notifications. Notifications can quickly inform you of an import that failed, so you can react more quickly.

Save and Load Packages

Use the Data Import Package workspace when import data needs to go through different test phases. For example, you may first import data to a test system and then on a pre-production system before importing it in the production system.

To do so, first create the data import package on your test system. When you are done creating, validating, and importing the package, use the **Save Package to File** action to save it from your test system. This package can then be loaded into the pre-production system using the **Load Package from File** action. Since this package is already validated and imported on the test system, there is a high chance that it can also be successfully imported on the pre-production system.

For example, you may go through these four systems/phases while making necessary adjustments to the import files along the way:

1. Create a new Package on **Development** system. Validate, import, and save it to file.
2. Load package file from **Development** system to **Test** system. Validate, make changes, import, and save it to file.
3. Load package file from **Test** system to **Pre-Production** system. Validate, make changes, import, and save it to file.
4. Load package file from **Pre-Production** system to **Production** system. Validate, make changes, import, and possibly purge.

Save as Template

Save as Template Procedures

Sometimes you might want to re-use the package structure for a new data import package. To do so, save the data import package in Skeleton Package Content format. This saves the underlying structure that defines the import programs and the order in which they should be imported. This saved structure can then be loaded into the new data import package. Instead of creating the import files manually, you have the option to load the package from an existing file package template instead.

Purge Package

Purge Procedures

The **Purge Package** action removes all the import data (import files and associated log files) from the system that are associated with this data import package. It does not automatically delete the data import package itself, however. Besides saving the package in Skeleton Package Content format, you are no longer able to make any changes to it. Note that after purging, the checksum information, as well as the import file names, sizes, and other information remains on the data import package for tracking purposes.

Best Practice: The data import packages often contain sensitive data about users and employees. Once the package has been successfully imported, it is best practice to run the **Purge Package** action.

Purge after you have imported into your production system, and then purge from all the environments where you imported previously (such as Test and Development). Purge from each environment manually, or schedule a background task to purge imported packages.

Data Import Packages Procedures

Data Import Packages Workflow

Create Data Import Package

You can create new data import packages via two methods:

- **From scratch**—Use for first instance or if no current data import package exists.
- **From import file**—Use when you have a package with many lines and you know which import programs you want to include in the package, and in what order. Additionally, you can choose to bulk attach import files to any number of package lines as you create the package.

Prerequisite: Before you begin, you must extract data from your old system and build import files to import to Maconomy. This step is out of scope of this section. These instructions pertain only to creating a new data import package, and not creating the data.

Create Package from Scratch

To create a new data import package from scratch:

1. In Workspace Client, go to **Setup » Data Import Packages.**
2. Click 

. A wizard displays.
3. Complete the following fields on the wizard:
    - **Name** — Enter a name that is representative of the new Data Import Package, such as London Office Vendor Invoices. This name must be unique per package. However, it can be changed after creation.
    - **Description** — Enter a brief description of the package that will be meaningful to you and others.
    - **Access Level** — Set the access level as needed.
    - **Deletable** — Select this box to indicate that the package can be deleted after creation.
    - **Email To** — Enter the email address(es) which must receive notification of import package progress.
    - **Email On** — Select Validation Finished to send an email regarding validation to notify you if the validation is successful or not Select Import Finished to send an email regarding the full import to notify you if the import is successful or not. You can select both options if needed.
4. Add package lines in the Data sub-tab.
5. Validate (or simulate validation) prior to import.

Create Package from An Import File

To create a package from a master import file:

1. Manually create a Data Import Package import file. This import file can only contain these line types:
    - Format (must contain the field **ImportProgram**)
    - Create (with or without specifying the keyword *create*)
    - Comment

2. If you want to bulk attach the import files during creation, include the field **AttachImportFileName** in the import file.

   Example import file for the Import Data Package:

```
DATAIMPORTPACKAGELINE:FORMAT IMPORTPROGRAM  Responsible    EmailTo
       AttachImportFileName

DATAIMPORTPACKAGELINE ImportSystemparameter User1   uone@company.com
       sysparametersstart.txt

DATAIMPORTPACKAGELINE ImportJobParameters   User2   utwo@company.com
       jobparametersdata.txt

DATAIMPORTPACKAGELINE ImportExchangeRates   User3   uthr@company.com
       exchangeratetables.txt


DATAIMPORTPACKAGELINE:FORMAT IMPORTPROGRAM

DATAIMPORTPACKAGELINE:create ImportJobParSelectionRules
```

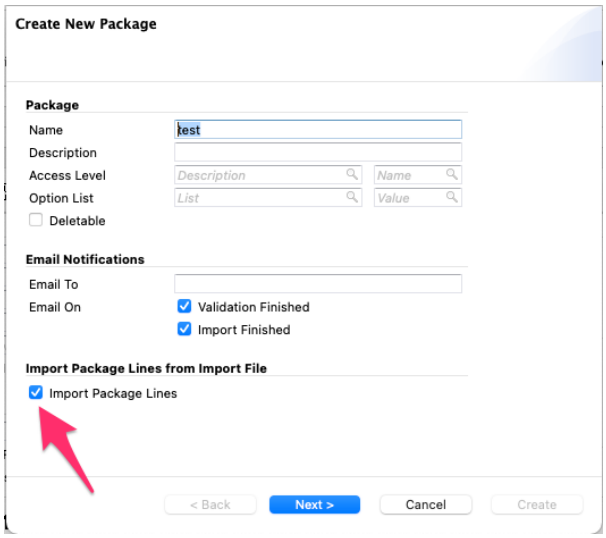
3. In Workspace Client, go to **Setup » Data Import Packages.**

4. Click 

. A wizard displays.

5. Complete the following fields on the wizard:

   - **Name** — Enter a name that is representative of the new Data Import Package, such as London Office Vendor Invoices. This name must be unique per package. However, it can be changed after creation.
   - **Description** — Enter a brief description of the package that will be meaningful to you and others.
   - **Access Level** — Set the access level as needed.
   - **Deletable** — Select this box to indicate that the package can be deleted after creation.
   - **Email To** — Enter the email address(es) which must receive notification of import package progress.
   - **Email On** — Select Validation Finished to send an email regarding validation to notify you if the validation is successful or not Select Import Finished to send an email regarding the full import to notify you if the import is successful or not. You can select both options if needed.
   - Select **Import Package Lines**.



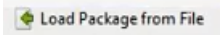
6. Click **Next**. Another wizard displays, where you can choose to attach files in a bulk action during the package creation.
7. Select **Attach Import Files to Package Lines**, and click **Create**.
8. Select **Import Package Lines**, and click **Next >**.
9. In the File Chooser, browse to select the master import file created in step 1.
10. If you have chosen to attach import files to the package lines, you are prompted with another File Chooser. Select any number of files to attach, including zip files containing the import files you want to attach (if the file is contained in a zip file, it is extracted and attached).

    The package is created with the package lines from the master import file. If you chose to attach import files to package lines during creation, all package lines which have a file name specified in the field **AttachImportFileName**, will get the file with that name attached as import file, if such a file is found among the files selected (or inside a zip file that was selected) in the File Chooser.

    For example, if you have an import file **exchangeratetables.txt** to attach to the package line with the import program **ImportExchangeRateTables**, then you put the value **exchangeratetables.txt** in the field **AttachImportFileName** on the line with the import program **ImportExchangeRateTables**. When you later select files to attach in the file chooser, you either select the file **exchangeratetables.txt** directly, or you select a zip file that contains the file **exchangeratetables.txt**. In both cases the file is attached to the package line.
11. After the package is created, a log file is generated which contains a summary of the import and details of the bulk attachment operation. You are prompted with another file chooser. Browse to select where to save the log file and click **Save**.

Load an Existing Data Import Package

To load an existing Data import package from an .mip file:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. Click 

. The Load Package from File wizard displays.
3. Select which load package format to use, such as Full Package Content to include all the original lines and files.
4. If needed, complete the additional fields:

- **Set Access Level** — Add the access level needed for this import package file, if any. Use this to field override the access level in the import package you are trying to load or assign one if there is none on the package that is being loaded.
- **Load *Email To* Information from Package** — Select if you must also include the same email notification information. Deselect if this information is not relevant to this company.
- **Clear *Import As* if user not present in system** — Select this field to clear a user and allow yourself to load the import package.

  For example, a user specified on the Test system may not exist on the Pre-Production system, which would generate an error. To avoid this situation, either create the user on the Pre-Production system before loading, or select this field.
- **Deletable** — Select to indicate that this import package can be deleted after its creation. If **Deletable** is not selected, the package can still be purged, but not deleted.

5. Click **Load**. A dialog displays a file chooser to allow you to choose a file with a *.mip extension.
6. Select the .mip file and click **Open**.

Add Package Lines to a Data Import Package

Once a package is created or loaded, package lines can be added in the Data sub-tab. Add lines in three ways:

- Manually
- Load lines from an existing package
- Load lines from a Master Data Import Package Import file

When lines are added by loading them from an existing package or an Import file, they are placed after the last current line in the table, in the order they appear in the package or import file.

Add Lines Manually

To add lines manually

1. In Workspace Client, go to **Setup » Data Import Packages » Data**
2. Add table lines with:
   - **Import Program** — Select the Window Name of the existing Maconomy import program to include in this import, such as Import Vendors.

     Add lines and select import programs to include in this import package as needed, taking care to add them in the order you wish to import them.

     > **For example**, import Employees prior to Time Sheets, as Time Sheets has a dependency on Employees.
   - **Attach Import File** — Click in the field on each line to open a dialog containing text files from the Import Program selected above, and select the related text file.

     > **For example**, for the ImportEmployees Import Program line, select Employees.txt.
   - **Import File** — This field automatically populates with the txt file selected by the **Attach Import File** action for this line.

Load Lines from An Existing Package

To load lines from an existing package:

1. In Workspace Client, go to **Setup » Data Import Packages » Data**
2. Select **Load Lines**.
3. Select **Load From Existing Package**. Click **Next.**
4. Select the Package Format to load. Click **Load.**

   A File Chooser will appear. Select the *.mip file to load lines from.

Load Lines from a Data Package Import File

To load lines from a data import package import file:

1. In Workspace Client, go to **Setup » Data Import Packages » Data.**
2. Select **Load Lines**.
3. Select **Load from Import File** as the method to load package lines. Click **Next**.
4. Select whether or not to attach import files to the lines as they are created.
5. Click **Create**.
6. A File chooser will appear, where you can select a master import files to import from.
7. If you chose to bulk attach import files to the lines, you will immediately be promptet with another File chooser where you can select any number of files to attach (in the same way as when you create a package from an import file)
8. A log file is generated, and you will be prompted to save it in a location of your choice.

Bulk Attach Import Files to Package Lines

The bulk attachment action is available by itself in the table part of the workspace. If you have specified the name of the import file to attach on one or more package lines, you can use the **Bulk Attach Import Files** action to attach the files to those lines.

To bulk attach import files to package lines:

1. Go to **Data Import Packages » Data sub-tab.**
2. Specify the name of an import file to attach on one or more package lines in the field **AttachImportFileName**.
3. Click Bulk Attach Import Files to attach the files to those lines You will be prompted with a File Chooser where you can select any number of files to attach.
4. A log file is generated, and you will be prompted to save it in a location of your choice.

Validate Packages

To validate the import lines:

1. Go to **Data Import Packages » Data sub-tab.**
2. Select the lines to validate.

- **To select lines one-by-one** select the **Marked for Validation** column for the line.
- **To select multiple lines at once** click the **Mark/Unmark** dropdown, and select **Mark All** to select all lines; **Mark to Here** to select lines prior to the current line; or **Mark from Here** to select lines after the current line.

Note: Selected lines display in bold font.

3. Click Validate Package. The activity and status lines automatically populate for all lines under validatation. Note also that the color updates to reflect the status. See the Legend for details.
4. Review **Validation Status**:
   - Status of Passed (green font) is validated and ready for import.
   - Status of Failed (red font) has errors as must be corrected prior to validation or import.
5. Identify and correct any errors in Failed lines.

   When all lines are in Passed status, the package is ready to import.

Simulate Import

To simulate the import:

1. Go to **Data Import Packages » Data sub-tab.**
2. Select a line in the table with an import file attached to it.
3. Expand the sliding panel on the right and select the Advanced tab.
4. Click **Simulate Import**.

After the simulation is complete, a text file displays a list of errors.

Import Packages

To import data package files:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. In the Mode island, under Validation, click **Switch to Import**. The mode switches to Import. Import actions appear, as does a Marked for Import column on the Data sub-tab.
3. Select the lines to import.
   - **To select lines one-by-one** select the **Marked for Import** column for the line.
   - **To select multiple lines at once** click the **Mark/Unmark** dropdown, and select **Mark All** to select all lines; **Mark to Here** to select lines prior to the current line; or **Mark from Here** to select lines after the current line.

   Note: Selected lines display in blue bold font.
4. Click Import Package. An Import Data Package wizard displays.
5. Complete the following fields on the wizard:
   a. **Import As (Default)** — The user specified in this field will be used for importing those package lines where an **Import As** user is not specified.

b. **Import All Lines Using Default User** — Select this field to import all lines as the user specified in the **Import As (Default)** field regardless of whether there is a user specified in the **Import As** field on the package line or not.

   Note: If there is no user specified in step b, then the default background task user is used if there is no user specified in the **Import As** field on the package line AND there is no user specified in the **Import As (Default)** field on the package.

   Additionally, the **Import All Lines Using Default User** functionality overrides if there is a user specified in the **Import As** field in the sub-tab or if a user is specified as a default background task user.

c. **Due Date** — You can schedule a validation/import date in the future. For example, you may want the import to start on 14 September, which is a Saturday when there are no users on the system.

d. **Due Time** — You can schedule a validation/import time for the date chosen above. For example, you can set the import starting on Saturday to begin at 8 a.m.

6. Click **Import Package**. The activity and status lines automatically populate for all lines to import. Note also that the color updates to reflect the status. See the Legend for details.

7. If any lines fail, correct errors. Otherwise, the **Status** for the data import package updates to indicate **Imported**.

8. If you intend to use this package again, you can use the action **Save Package to File** to Create an Import Package Template.

Correct Errors

Maconomy stops the import at the point of a failed line, due to line dependencies. Failed lines are indicated by red font, red highlight, and the **Validation Status** of **Failed**. To resolve errors, review the Validation Log or Import Log, depending on the mode you are in.

To correct errors:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.

2. In Validation Log or Import Log (depending on which mode you are in) click View for the Failed line. A text log file opens, flagging the error lines.

3. Identify the errors, and investigate as needed.

4. On your local drive, open the related text file, and make necessary corrections.

5. Save the file.

6. Reattach the file to the import line. Click the **Attach Import File** field on each line to select the related text file. The **Import File** field automatically populates with the txt file selected by the Attach Import File action for this line.

7. Proceed with the Import or Validation.

   Note: We retain the logs for future reference if needed, though they are now labeled “Outdated.” They are overwritten with a new log when you re-validate the line.

Purge Packages

This action deletes all the import files related to this package as well as any validation and import logs and locks the package from further use.

To purge an import package:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. Select the package in the filter **List of Data Import Packages**.
3. In the Data Import Package tab, click Purge Package. The Mode updates to Purged, and fields become locked.

   Note: The **Remarks** field remains editable after purging so that you may add any notes or details as needed.

Alternatively, you can delete the package, if the Deletable field is selected in the Advanced assistant on the tab.

1. If you have proper permission (via **Make Packages Deletable**), go to the Advanced assistant on the tab and ensure the **Deletable** field is selected.
2. Click **Delete**.

Create an Import Template File

If this is an import setup that you will have to run again at numerous company sites, you can save the Maconomy Import Package File to a single file for reuse. This action archives all the files in the package into a single file.

Note: Template files have the extension **.mip** (Maconomy Import Package).

To create an import template package file:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. Click Save Package to File.
3. In the dialog that displays, choose to save the file in one of the following formats:
   - **Full Package Content** — Select to include all the information that is included in the import package, including all the information in the Data Import Package tab, and Data sub-tab.
   - **Template Package Content** — Select to reuse format lines in import file, which define the columns to specify in import file and load without loading the actual data. This allows you to reuse the template file without reusing the data itself. It includes Import Files (.txt files) but the text files contain no data.
   - **Skeleton Package Content** — Select to reuse only the structure of the files. This setup contains no import files, and only includes the lines that define the import programs to use as well as the structure to use. With this format, users can attach their own import files.
4. Click Save Package to File.
5. On the dialog that displays, confirm or update the name of the package.
6. Click **Save** to save on your local drive.

Setup Instructions

To enable this feature:

1. Go to **System Setup » Parameters and Numbers » System Parameters**.
2. Double-click the **Use Internal Names in Import Dialogs** parameter (see description below).
3. On the System Parameter sub-tab, select the **Use Internal Names in Import Dialogs** check box.
4. Click **Save.**

Install / Upgrade Considerations

Note that the **Use Internal Names in Import Dialogs** system parameter is enabled after the upgrade, so that the Internal Names field is also selected in all the import dialogs.

System Parameter for Import Programs

The **Use Internal Names in Import Dialogs** system parameter enables Maconomy to automatically include internal names in import criteria as a default setting when you open any import dialog box. This system parameter is enabled by default, but can be disabled in the System Setup.

See Data Imports Packages Troubleshooting for solutions to common issues.

Standard Import Programs Overview

NOTE: While the Standard Import Programs continue to function as always, new Data Import Packages functionality streamlines the import process.

This section describes the use of import programs for Maconomy 2.0+. This section describes the central concepts involved in the import process and explains the possibilities offered by import programs. In addition, this section 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 section 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.

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.

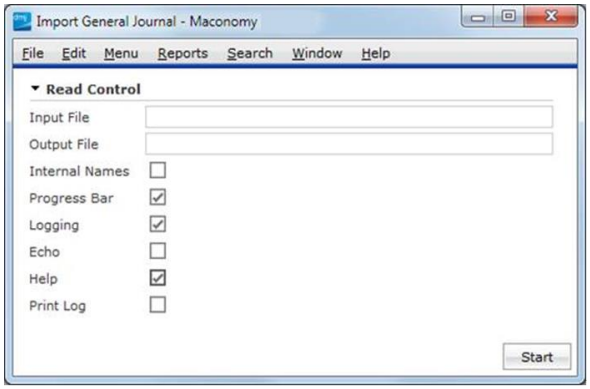
Activating Help

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.

> **Note:** 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.

> **Note:** 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.

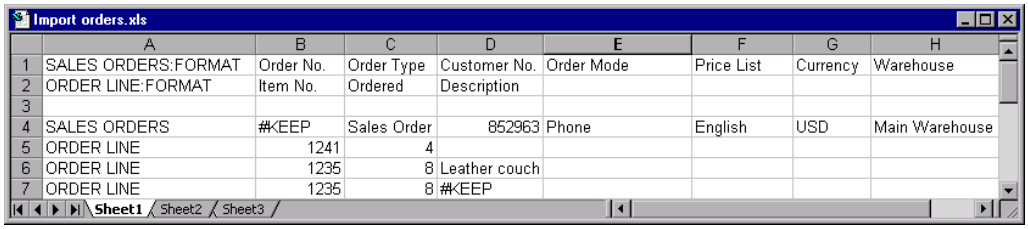
To activate PrintFormat, complete the following steps:

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:



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

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”.

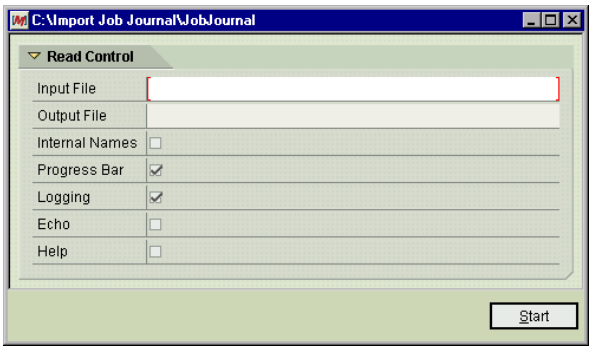
In addition to the standard functionality, some import programs offer a number of functions specific to the program in question. For instance, the Import Orders program allows the approval of sales orders, which is a function that is only relevant when working with sales orders. The functions that are 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
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

Parameter Name	Functionality	Default Value
UseInternalNamesVar	Corresponds to the **Internal Names** field in the import dialog, as described in "Running Import Programs Manually." The value 1 corresponds to the **Internal Names** field being selected in the import dialog. The value 0 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 corresponds to the **Progress** field being selected in the import dialog. The value 0 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 corresponds to the **Logging** field being selected in the import dialog. The value 0 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 corresponds to the **Echo** field being selected in the import dialog. The value 0 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 corresponds to the **Help** field being selected in the import dialog. The value 0 corresponds to the field not being selected in the import dialog.	0/"false"
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, 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.	Blank

Parameter Name	Functionality	Default Value
	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.	
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>
```

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.

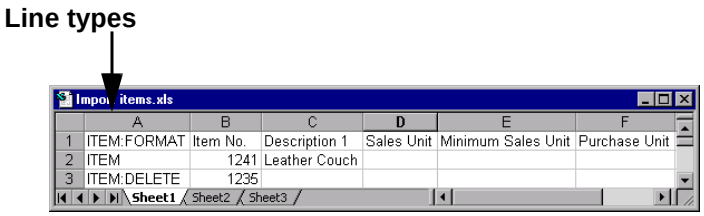
Entries

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
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, Đ, 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.

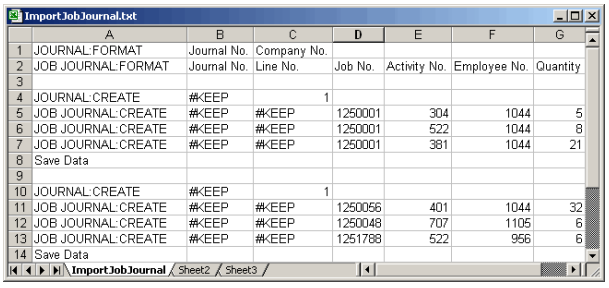
Data Commitment to Database and Errors During Import

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.



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 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

Type	Example	Description
		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.

Import users.xls

	A	B	C	D	E	F
1	USER:FORMAT	Username	Valid From	Valid To	Employee No.	Company No.
2	USER DIALOG GROUP:FORMAT	Username	Line No.	Group Name		
3						
4	USER:CREATE	John Smith	01.01.03	31.12.03	12	1
5	USER:CHANGE	Jim Pooley	01.01.03	31.12.03	#KEEP	2
6	USER	Chris Slocombe	01.03.03	01.06.01	24	1
7	USER DIALOG GROUP	#KEEP	#KEEP	Sales		

Sheet1 Sheet2 Sheet3

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.

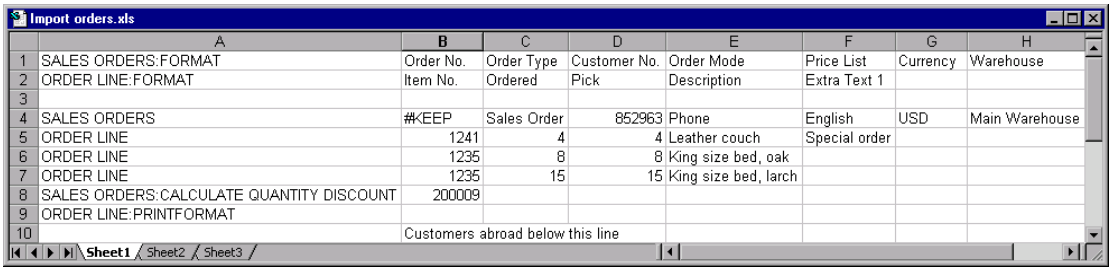
Format Return Lines

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.

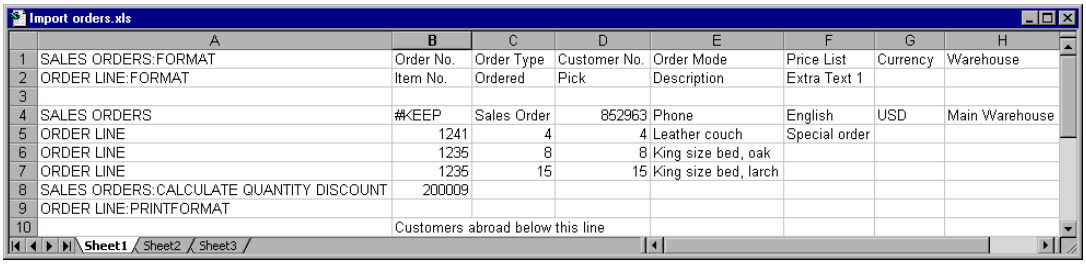
Some parameter sets contain only one parameter field, while others contain a number of fields. Therefore, you must always specify a format line for parameter sets before an actual parameter line can be imported, regardless of whether the parameter set consists of only one field. Format lines for parameter 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.

Field is a Key Field	Field is Mandatory and Open in Create	Field is Maintained by Maconomy	Result
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.

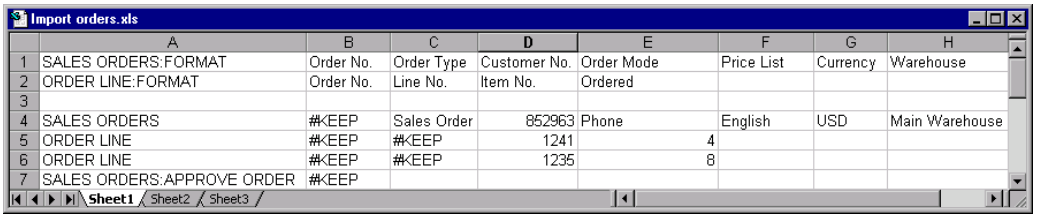
Using #KEEP to Use Information from Previous Entry

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.



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.

EXTEND YOUR SYSTEM

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.

Talent Management

The Talent Management integration enables Maconomy customers to use Talent Management to manage recruitment and other HR processes while utilizing Maconomy for their ERP solution.

See the *Deltek Maconomy Integration with Talent Management* document for more details.

CRM

The CRM integration enables Maconomy customers to use Deltek CRM to manage contacts and opportunities while utilizing Maconomy for their ERP solution.

See the *Deltek Maconomy Integration with CRM* document for more details.

People Planner

Deltek's People Planner is a separate Deltek offering that combines graphical project and resource planning for your entire resource pool and project portfolio on a company-wide scale.
You can integrate People Planner with any of the three Maconomy solutions (standard, CPA, or PSO) to combine its graphical planning capabilities with the advanced project economy management features of Maconomy.
The suite of People Planner programs includes the following:

- **Deltek People Planner** — Sometimes called the "People Planner Client," although it is not strictly a client because there is no server. It is sometimes just called "People Planner," which can cause some confusion as to whether the reference is to the Deltek People Planner product or the entire suite of programs.
- **People Planner Admin Tool** — A tool used to perform the initial setup and maintenance of People Planner.
- **Deltek MyPlan** — A light web page client.
- **Deltek People Planner API** — A web service used to integrate People Planner with the Deltek Maconomy software.
- **Deltek People Planner Outlook Web Service** — A web service used to integrate People Planner with Microsoft® Exchange® and Outlook®.
- **Deltek People Planner Outlook AddIn** — An add-in for Outlook. The People Planner Outlook Web Service, the People Planner Service, and this add-in form the components of the integration between People Planner and Exchange/Outlook.
- **Deltek People Planner Service** — A Windows service that is responsible for scheduling tasks.

People Planner Admin Tool

Use the People Planner Admin Tool for the initial setup and maintenance. This tool enables you to create and maintain the People Planner database, add People Planner users, and create the configuration files used by the People Planner client.

To test the integration using the Test Integration Settings in the People Planner Admin Tool, you must provide the People Planner integration user with temporary Read access to the System Parameters container.

Integration with Microsoft Outlook

The integration with Exchange/Outlook enables People Planner to use Outlook, instead of the People Planner client, as your personal planning tool.

Integration with Maconomy

You can use People Planner as a stand-alone resource planning tool; however, it can also be integrated with Maconomy. This integration enables the People Planner client to import master data from Maconomy that is relevant for project and resource planning.

The Resource Management workspace allows access to the People Planner Capacity Overview page from within the Workspace Client. The workspace enables you to access and control data based on the People Planner access you are granted.

Enter the People Planner URL in the system parameter, **URL for Capacity Overview**, otherwise, the Resource Management workspace is hidden.

People Planner can be integrated with Maconomy X1 Service Pack 16 or later; or with Maconomy 2.0.

To enable the Resource Management Workspace:

1. Go to **Setup » System Setup » Parameters and Numbers » System Parameters**.
2. Double-click the **URL for Capacity Overview** parameter.
3. In the System Parameter sub-tab, enter the People Planner URL in the **URL for Capacity Overview** field.
4. Restart Maconomy.

Fields / Descriptions

Field	Description

URL for Capacity Overview	The URL of the People Planner Capacity Overview page. The general form is:
	`http://<server>:<port>`
	`/PeoplePlannerWebComponents`
	`/AssignResourcePage.aspx`

Periodic Job Budget System Parameters

The following system parameters configure how the **Update Periodic Job Budget from People Planner** and **Send Job to and Update Periodic Job Budget from People Planner** actions behave.

- **Update Periodic Job Budget Planned Dates on update from People Planner** – updates the Planned Starting Date and Planned Ending Date when Update Periodic Job Budget from People

Planner or Send Job to and Update Periodic Job Budget from People Planner actions are invoked.

- **Submit Periodic Job Budget on update from People Planner** – submits the budget when **Update Periodic Job Budget from People Planner** or **Send Job to and Update Periodic Job Budget** from People Planner actions are invoked.

The system parameters are enabled by default. Deselect the system parameter to disable it.

To configure the Periodic Job Budget System Parameters:

1. Go to **Setup » System Setup » Parameters and Numbers » System Parameters**.
2. Double-click the **Update Periodic Job Budget Planned Dates on update from People Planner** parameter.
3. In the System Parameter sub-tab, select the **Update Periodic Job Budget Planned Dates on update from People Planner** field.
4. Click on **Show Filter List**.
5. Double-click the **Submit Periodic Job Budget Planned Dates on update from People Planner** parameter.
6. In the System Parameter sub-tab, select the **Submit Periodic Job Budget on update from People Planner** field.

Fields / Descriptions

Field	Description
Update Periodic Job Budget Planned Dates on update from People Planner	Updates the job budget planned dates after clicking the **Update Periodic Job Budget from People Planner** or **Send Job to and Update Periodic Job Budget from People Planner** actions. Enabled by default.
Submit Periodic Job Budget on update from People Planner	Automatically submits the budget after clicking the **Update Periodic Job Budget from People Planner** or **Send Job to and Update Periodic Job Budget from People Planner** actions. Enabled by default.

Capacity Overview Tab and the People Planner Tab

The Resource panels user interface are renamed Capacity Overview and People Planner in the Jobs workspace:

- In **Jobs » Budgeting » Budget**
- In **Jobs » Budgeting » Periodic Budget**
- In **Jobs » Budgeting » Job Planning**
- In **Jobs » Progress Evaluation**

The following labels are updated:

Old User Interface Label	Updated User Interface Label
Capacity Overview	Resource Assignment
People Planner	Resource Booking

Additional Fields Sent to People Planner

In the Resource Assignment tab, the following additional fields required by the **Assign to Budget** action are sent to People Planner:

- Employee Category No.
- Line Type
- Budget Type
- Job Number

Changing any of these fields in an open job automatically sends the updates to People Planner.

Create Standard People Planner Integration User

The PSO Solution configuration is enhanced to improve support for the People Planner integration. The access granted to the People Planner Integration user (using the import files) meet the minimum requirements for integration to function properly.

However, in order to test the integration using the Test Integration Settings in the People Planner Admin Tool, you need to provide temporary Read access for the People Planner Integration user to the System Parameters container.

Periodic Job Budget Action

The **Send Job to and Update Periodic Job Budget from People Planner** action supports the previously-used action sequence to send the job to People Planner and then update the Periodic Job Budget on the job.

The following are recommendations for usage:

- Use **Update Periodic Job Budget from People Planner** when the Periodic Job Budget has to be updated from People Planner but does not have to be updated in People Planner.
- Use **Send Job to and Update Periodic Job Budget from People Planner** when the Job Budget has to be updated in People Planner before the updates are retrieved for the Periodic Job Budget.

Integration with Web Client

The **Enable People Planner Web** company-specific system parameter in the Workspace Client determines whether People Planner-related components and functionality are made available in your web client application. The company of the user determines if the new People Planner related workspaces and actions are available for the user in the web client.

Setup Instructions

This section assumes that Maconomy-People Planner integration is already enabled. To enable People Planner in the web client:

1. In the Workspace Client, go to the System Parameters single dialog.

2. Search for **Enable People Planner Web** in the filter list.
3. In the System Parameter tab, select the **Enable People Planner Web** check box either in the card for all companies, or in the table for a specific company.
4. Click **Save**.

By default, this system parameter is disabled across all solutions, whether upgrading or setting up a new installation.

Note: The People Planner-related actions and wizards in in the web client are only available for jobs with the **Use Imported Budget Line Resource Allocation** job parameter attribute enabled.

System Architecture

People Planner is based on Microsoft technology and is, in essence, a .Net application. The users can choose a Windows or a web-based client. The Windows client offers significantly more functionality over the web client. The Windows Client is primarily used by Resource Managers and Project Managers, whereas the web client is used by the individual employees (resources). It is possible to configure a People Planner installation where everybody uses the Windows client, allowing the web client to be optional.

Warning: It is not possible to deploy a People Planner solution without the Windows client, because all of the resource management functionality resides in that client.

People Planner uses Microsoft's SQL Server for data storage and communicates directly with this from the client.

Administrative users log in to People Planner with a local account on the People Planner database server. Users get logged in automatically through their Windows user account.

When configured with a Web client interface, this is handled by Internet Information Server (IIS).

While People Planner can operate as a standalone solution, it typically interacts with the following external systems:

ERP — Projects are usually created in the ERP system, such as Deltek Maconomy, where they are assigned appropriate metadata (Project Number, and so on). The project's baseline budget is also established there. After the project and the baseline budget are created in the ERP back end, they can be copied into the People Planner database, so that the resource management workflow can begin (resource assignments, and so on). People Planner can retrieve the information from the ERP system by reading a CSV/XLS file generated by that system, or through direct SQL access to the ERP database. Both of these methods can run on a scheduled basis through a scheduler tool.

People Planner can also deliver information back to the ERP system, such as budget updates, changes to project completion dates, and so on.

Microsoft Exchange — When People Planner is integrated with Exchange, users can see People Planner bookings (made by the Project Managers) in their Outlook appointments. The integration with Exchange is done via the EWS Exchange Web Service.

Microsoft Excel — People Planner can export data to Excel for further manipulation.

Microsoft Project – People Planner can export a project to a Microsoft Project compatible file and also import Microsoft Project files as new People Planner projects.

BI — People Planner can deliver data to a Business Intelligence (BI) system for further analysis.

This is typically done on a database level.

The following information describes the logical tiers that are depicted in the following illustration.

People Planner's tiers are all logical tiers, which can be collapsed or allocated to as many physical servers as needed by your firm. Those determining factors are discussed in more detail as part of the sizing planning section of this document and can include user concurrency, location, access expectations, and business needs.

E-Invoicing with Pagero

Maconomy allows electronic invoicing through an integration with Pagero, a third-party provider of e-invoice products. Their solution supports e-invoicing in many countries and allows mapping to local formats, including the German XRechnung format.

Originally intended to address a German statutory requirement, any company can use this functionality as long as they meet the following requirements:

- They reside in a country that uses the PEPPOL BIS 3.0 format.

  Some countries may require country-specific updates to the mapping.
- They have a Pagero account, and Pagero supports e-invoicing in their country.

Note that only customer invoices sent to Pagero are covered (with the exception of on account invoicing of sales orders and subscription orders, which will be supported in a future release). To date, the Workspace Client is unable to process vendor invoices.

Setup Instructions

E-invoicing is enabled on the level of the company customer, and is available on all three solutions: standard, CPA, and PSO.

The following procedure assumes the following prerequisites are in place:

- You already have a Pagero account, and the accompanying login credentials.

  For information on account creation and setup, go to: https://www.pagero.com/
- You already added your customers to your directory in Pagero Online.

To enable e-invoicing in Maconomy:

1. Open the server.ini file in the Coupling Service configuration folder.

   > **Note:** If you are a customer on Maconomy Cloud, you should skip steps 1-3 of this procedure, and file a support case with Deltek Cloud Operations. You will need to provide your Pagero client ID and client secret.

2. At the end of the file, add the following information based on your Pagero account details:

   ```
   url.pageroclientid = <your Pagero client ID>
   url.pageroclientsecret = <your Pagero client secret>
   url.pageroauthurl = <the authentication URL for the Pagero API>
   url.pagerofileurl = <the file URL for the Pagero API>
   ```

3. Define system natures by adding the following text immediately before the Pagero account information you specified in the previous step.

Using the following format, add as many lines as you need for the number of test/production systems your company maintains.

```
system.nature.<system shortname> = <system nature>
```

4. Open the Workspace Client.
5. Go to **Single Dialogs » Integrations » Pagero Setup single dialog**.

   If you prefer to use the Pagero Setup workspace for the next steps, go to **Setup » Pagero Setup workspace » Pagero Setup**.
6. Enter values for the following fields:
   - **Pagero Username** – Enter the e-mail address used for your Pagero account.
   - **Production Nature** – If the current system is a production system, specify its corresponding production system nature as listed in the server.ini file.
   - **Test Nature** – If the current system is a test system, specify its corresponding test system nature as listed in the server.ini file.
7. Go to the **System Setup workspace » Parameters and Numbers**.
8. In the System Parameters filter list, double-click the **Enable E-Invoicing** system parameter.

   The Workspace Client closes the filter list.
9. In the System Parameter tab, select the **Enable E-Invoicing** check box, and then click **Save**.

   This allows you to enable the functionality on the following levels: company, customers, and company customer.

   When you enable this system parameter, Maconomy also enables the four standard mappings, one for each of the mapping types. These are automatically set as the default mappings, unless you create customized mappings and set those as the default.

   The four standard mappings are:
   - Job Cost Credit Memo
   - Job Cost Invoice
   - Order Credit Memo
   - Order Invoice
10. Restart the Workspace Client.
11. To set up the mapping between Maconomy data and fields in the PEPPOL BIS 3.0 format, go to **Single Dialogs » Integrations » Peppol Mappings single dialog**.

    If you prefer to use the Pagero Setup workspace for the next steps, go to **Setup » Pagero Setup workspace » Peppol Mappings**.
12. Select a standard mapping type from the filter list. The Workspace Client closes the filter list.

    Since this is a standard mapping type, the only customizations you can perform at this point are:
    - To enable/disable this as the default mapping for the mapping type.
    - To show/hide unused lines.
13. To create a customized mapping using the standard mapping as a base:
    a. On the Peppol Mapping Header tab, click the green plus sign icon (also known as the **New Peppol Mapping Header** action).
    b. Fill out the following fields:
       - **Mapping Name** (mandatory)

- **Mapping Type** (mandatory) – Specify the mapping type you want to use as the base for your customized mapping.
- **Default Mapping** – Select this check box if you want to use the customized mapping as the default mapping for the mapping type you specified.
- **Show Unused Lines** – Select this check box if you want to show unused lines in the Peppol Mapping Line sub-tab.

c. On the Peppol Mapping Line sub-tab, you can customize how Maconomy fields map to fields in the PEPPOL BIS 3.0 format. Specifically, you can utilize the following line fields:

- **Included** - Select/clear this checkbox to include/exclude the current PEPPOL field from the output xml.

  Note that some PEPPOL fields are mandatory and cannot be excluded.
- **Maconomy Relation Name** and **Maconomy Field Name** – Use these fields to specify which Maconomy database field value will be shown on the current PEPPOL field when the output xml is generated.
- **Text** – If you want the current PEPPOL field to display fixed text, use this field to specify that text.
- **Standard Calculation** – If you want the PEPPOL field to display a value calculated from the values of other fields in the Maconomy database, use this field to make that specification. Select a value from the drop-down. The options available in this drop-down will depend on what you specify in the **Maconomy Field Name** field.

  You can also enter descriptive text on the **Calculation Description** field.

  > **Note:** For more information about the PEPPOL BIS 3.0 standard, go to https://docs.peppol.eu/poacc/billing/3.0/bis/#_document_structure.

d. Click **Save**.

14. You can now enable e-invoicing and specify mappings at the various levels: company, customer, and company customer.

    Maconomy will begin to look for e-invoicing and mapping specifications at the lowest level (company customer). If nothing is specified at that level, it will follow the specifications at the next level (customer). If nothing is specified at that level, it will follow the specifications at the highest level (company).

    Note that you cannot enable e-invoicing (and make specifications) at a lower level if you have not enabled the functionality at the level immediately above it.

15. To set up e-invoicing at the company level:
    a. Go to the Companies workspace.
    b. Select a company from the filter list.
    The Workspace Client closes the filter list.
    c. Go to **Home » Information tab**.
    d. Under the E-Invoicing island, fill out the following mandatory fields:

    - **Enable E-Invoicing** check box
    - **Electronic Address**
    - **Electronic Address Scheme** – For this field, use the following link for reference:

      https://docs.peppol.eu/poacc/billing/3.0/codelist/eas/

    e. Under the E-Invoice Mappings island, fill out the following fields.

For each field, select any standard or customized mapping available for that mapping type.

- **Jop Cost Invoice Peppol Mapping**
- **Job Cost Credit Memo Mapping**
- **Order Invoice Mapping**
- **Order Credit Memo Mapping**

f. Click **Save**.

16. To set up e-invoicing at the customer level:
    a. Go to the Customers workspace.
    b. Select a customer from the filter list.

       The Workspace Client closes the filter list.

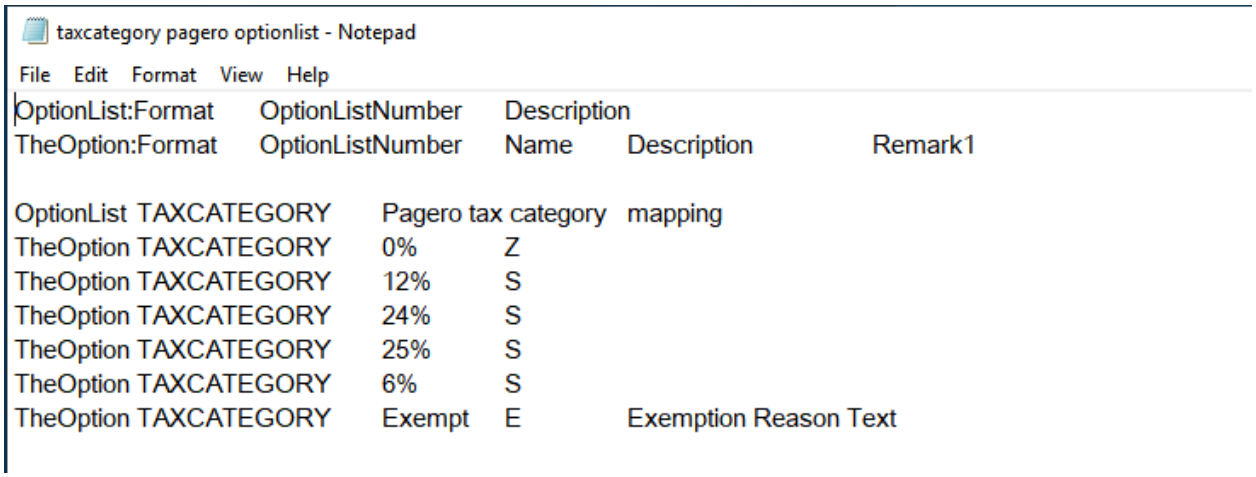
    c. Go to **Home » Information tab**.
    d. Fill out the mandatory fields under the E-Invoicing island.
    e. Fill out the fields under the E-Invoice Mappings island. For each field, select any standard or customized mapping available for that mapping type.
    f. Click **Save**.
17. To set up e-invoicing at the company customer level:
    a. While still in the Customers workspace, expand the Company Customers sub-tab for your selected customer.
    b. Select a company customer from the filter list in the sub-tab.
    c. Expand the Details sliding panel in the sub-tab.
    d. Fill out the mandatory fields under the E-Invoicing island.
    e. Fill out the fields under the E-Invoice Mappings island. For each field, select any standard or customized mapping available for that mapping type.
    f. Click **Save**.
18. You also need to ensure that your tax codes and tax tables are set up and mapped correctly, so that VAT is calculated on the outgoing invoice.

    Perform the following steps:

    a. Follow the procedure for creating an import file for the tax category option list.

       The format is the same for both on-premises and Cloud customers, but the contents of the import file depend on how the tax codes are set up on the system (varies from customer to customer).

       The following is a sample option list:



Note the following:

- The option list number should be **TAXCATEGORY**.
- For each tax code in your tax code table, add an option to the list where:
  - **Name** = the tax code name
  - **Description** = the tax category for that tax code.

    For the list of PEPPOL BIS 3.0 tax categories, refer to the following: https://docs.peppol.eu/poacc/billing/3.0/bis/#_vat_category_codes

    Remark 1 = If you use the E tax category (Exempt from Tax), use this column to enter the tax exemption reason text.

b. Import the import file to the Workspace Client.

c. Proceed with the usual procedures for tax setup.

Procedures for Addressing Common Errors

Resend an Invoice

You can resend an invoice by performing the following steps:

1. Go to the Show All Invoices single dialog.
2. Open the invoice you need to resend.
3. Click the **Send to Pagero** action.
4. Accept the warning that Maconomy displays.

Pagero-Generated Error

If Pagero has stopped an invoice due to an error in Maconomy, create a Customer Care case. Include the error message from Pagero and the invoice information.

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.

Translations

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++.

  Warning: “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. Mgrs.”.
- 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 online help and Maconomy Information Center**

  The online help provides detailed information about the functionality of the Maconomy system. The Maconomy Information Center is a hub for all Maconomy documentation at a single stand-alone link you can bookmark. Make sure you have it.

  See Maconomy Information Center for details.

- **Release Notes / Enhancements Guide**

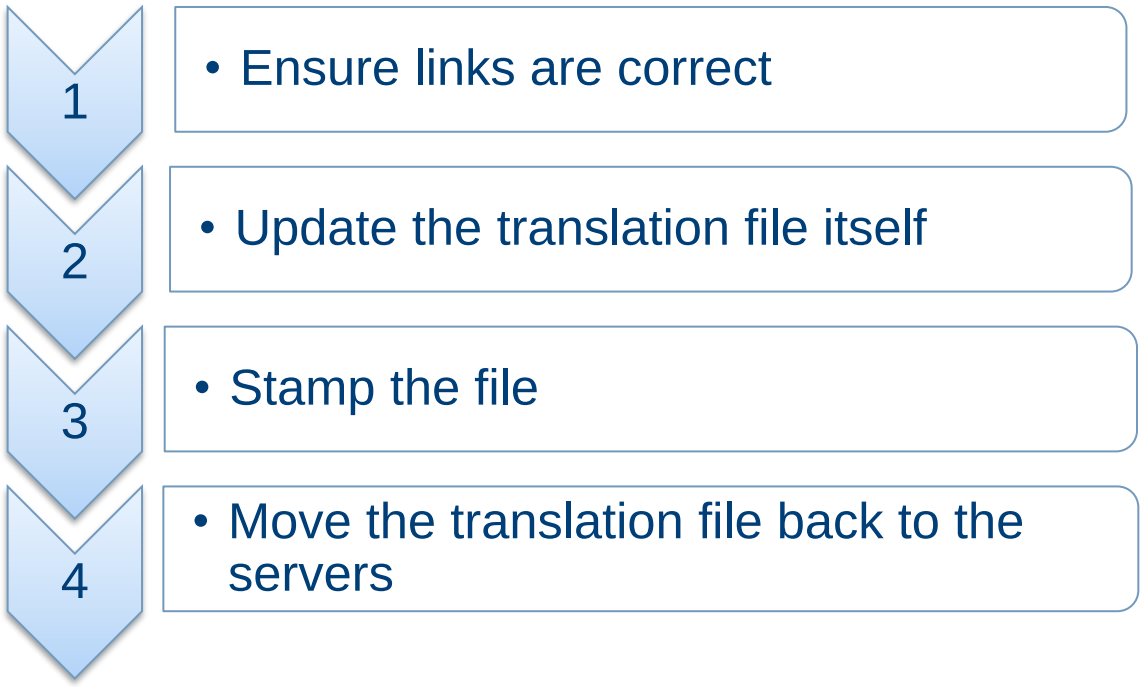
  The Release Notes and Enhancements Guide describe the new features introduced from one version of Maconomy to the next. By reading this information for the version whose missing list you are translating, you are acquainted with the functionality behind the strings in the missing list, which is a shortcut 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). 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.

Warning: 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.

Update the Translation Files

Translation files are kept in two locations:

- Application server
- Web server

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.

   Warning: 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.

   Warning: 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

To 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.

Attention: 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

> **Warning:** 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:
    a. 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.
    b. 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.

    > **Note:** 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.

> **Note:** 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 coupling services on the App server (under **Start > Services**).
3. Re-confirm that no users are logged on.
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**

> **Note:** 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.

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

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."

> **Tip:** 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.

Translation Procedures

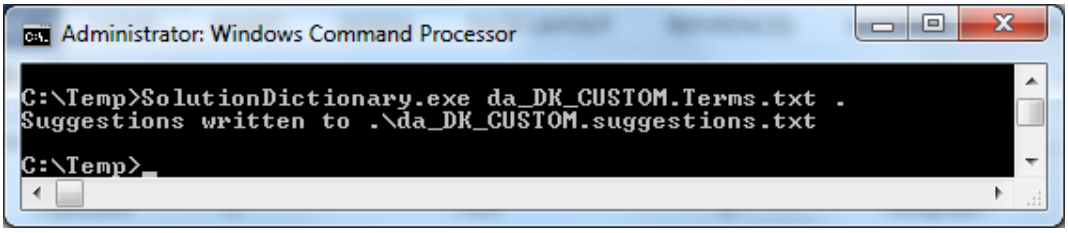
Use the Custom Program

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.

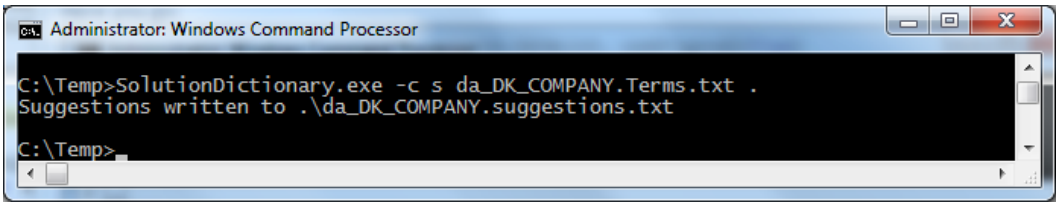
To use the program, complete the following steps:

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 Online Help** — The reference online help (formerly the Reference Manual) provides detailed information about the functionality of the Maconomy system.

- **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.

Automation and Customization

Most customer systems require automations and/or customizations. The following sections some of the tools/possibilities for customized solutions.

TROUBLESHOOTING

This section contains troubleshooting tips for various sections.

Maconomy Server

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.
- 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.
- For more information about performance, see "Performance" in this manual.

Background Tasks FAQs

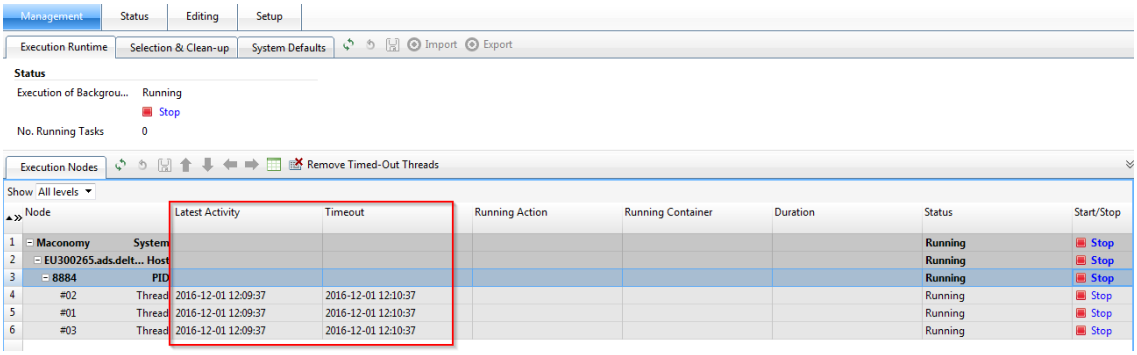
This section provides Frequently Asked Questions for Scheduled Background Tasks.

Is the Background Task System Running?

You need to check what background execution threads are running, and verify that they are “alive”.

To do that, go to **Set Up » Background Tasks » Management » Execution Runtime » Execution Nodes**.

This view shows all execution nodes “recently known” as well as their activity level:



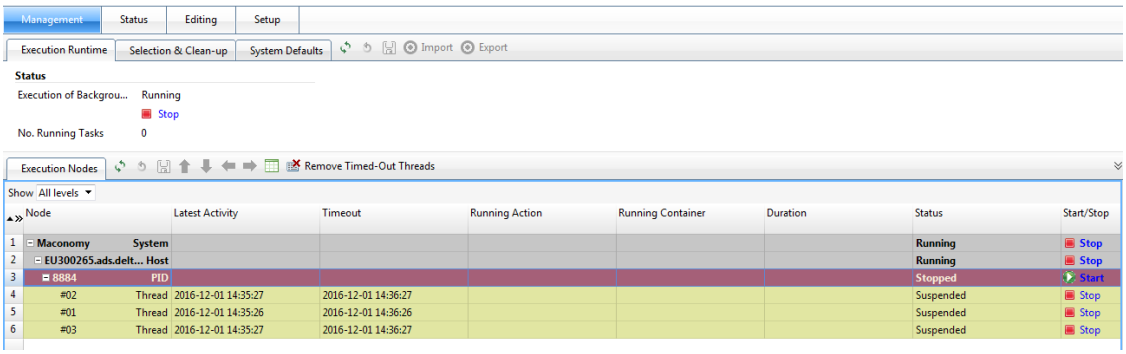
Each node of type “Tread” represents an execution thread that can run a task. The threads are grouped into running processes represented by the PID, grouped again under the host name of the computer hosting that process.

Every time a thread makes itself known (to query for background tasks) the “Latest Activity” time stamp is updated. Hence, using this field, you can see when a given thread was latest making itself known. The “Timeout” timestamp means that if the thread does not make itself known again before that time, the system may consider the thread as “timed out”, and may remove it from this view. If – at that time – the thread is believed to be running an actual background task, that background task will automatically be “aborted” by the system.

It is possible to “Stop” a given execution node. That means that no new background task will be assigned that node.

In the picture below, the node with PID 8884 is “stopped”, meaning that background tasks will not be assigned to any execution node belonging to this PID. The threads may still be rendered as “active” (their “Latest Activity” is updated), but no matter what background tasks will not be assigned to those nodes.

> **Note:** If all threads are Suspended/Stopped, no background tasks are processed!



I Need to Shut Down the Coupling Service – Should I Do Anything?

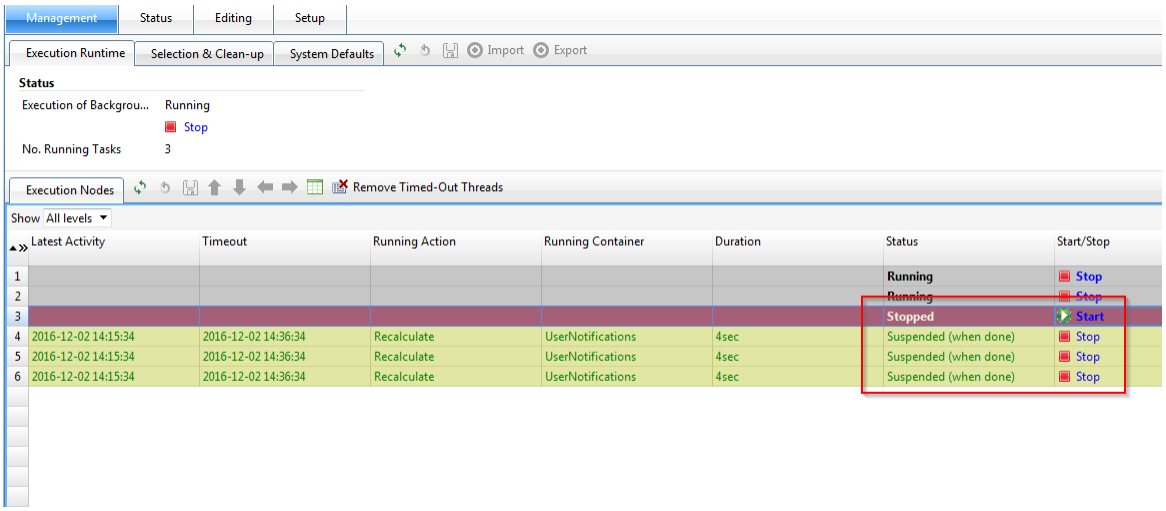
You don't have to do anything. But if the coupling service is executing a background task while you shut it down, that task will time out and be aborted by the system.

In order to avoid this situation, a "nice" behavior is to stop background task execution for the specific process ID, and once all underlying task threads are done executing tasks, stop the coupling service.

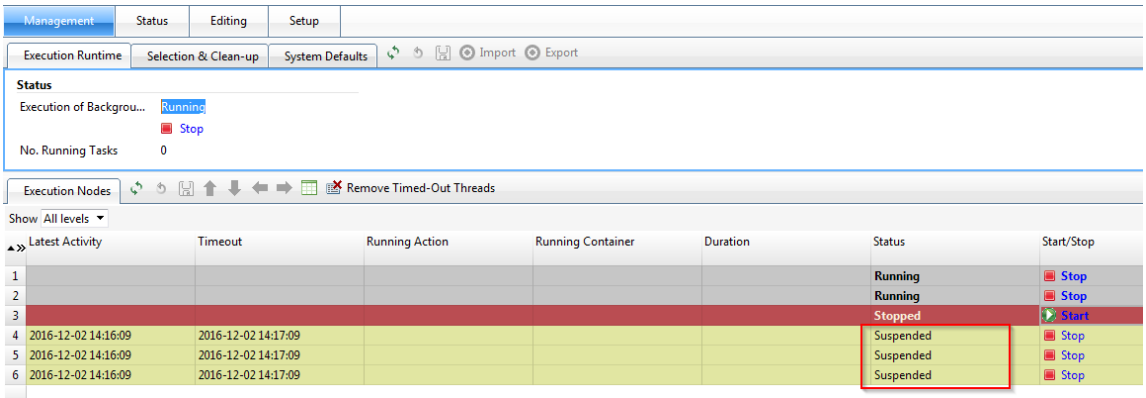
Note: you cannot get "damaged data" simply by shutting tasks down, but you may end up having tasks that are marked as failed, because they are aborted by the system. Such tasks can be rescheduled, but the decision to do so is a manual one.

You can stop execution of background tasks for specific processes in Set Up / Background Tasks, Management » Execution Runtime » Execution Nodes.

After pressing the "Stop" button for the process ID corresponding to the coupling service to be stopped, the view may change into something like (assuming that tasks were running):



By refreshing the view, you can see when the tasks are done:



Now that process can be stopped without leading to abortion of background tasks.

I Need to Copy the Production Database to Test – Can I Do So Safely?

When a database copy is done, the background tasks are also being copied (in their current state) since they are part of the data. For many tasks this is not a problem at all.

However, if some background tasks communicate with external systems (mail servers, file system, web services) it may be relevant to ensure that the production and test systems are configured to handle these

external resources differently. And/or to ensure that background tasks that should only run on one of these systems are annotated with a proper “nature”.

Natures are defined in the server`.ini` file. As are named file- and URL references as well as mail servers. You should look into the documentation in the `server.ini` file for these properties.

If this has been set up properly, database copies can be done without issues.

Alternatively, you may choose to disable background task execution for the short name to which to database is copied, and manually examine the pending background tasks/background task schedule rules before enabling background task execution for such systems.

What is the Status of the Background Tasks?

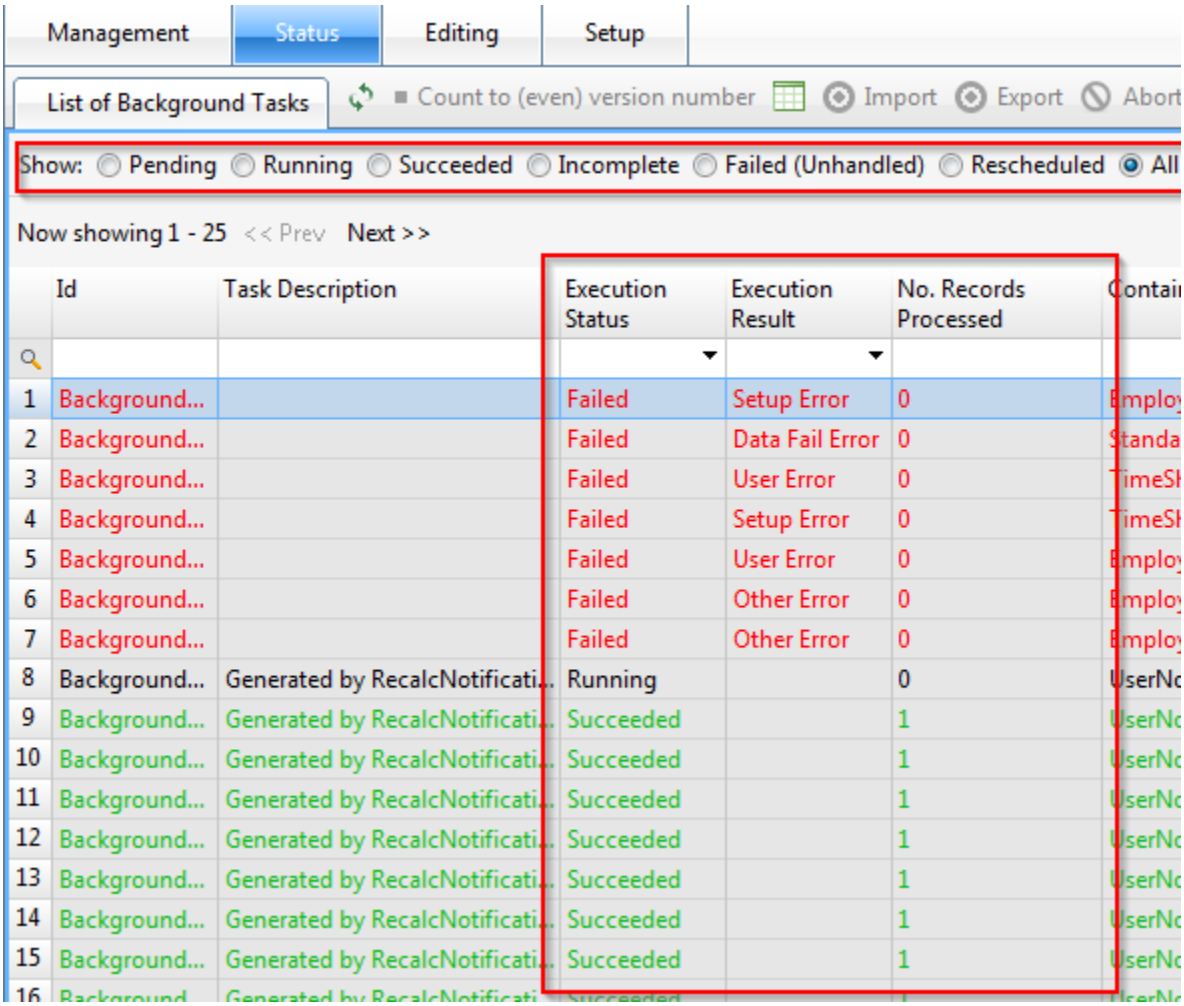
Background tasks can essentially be in one of the following states:

- Pending - The task is ready to be picked up for execution (when it’s active, it’s due time is up and there’s nothing to await)
- Running - The task is currently being executed
- Committed - The “logic” of the background task has succeeded and has been committed, handling of output documents has not been finalized yet
- Incomplete - After the task was “Committed” the handling of output documents failed for some reason. For example, it was not possible to send mails, or not possible to store files
- Succeeded - The task has succeeded: the “logic” has been done and committed, and all handling of output documents (if any) has been successfully completed.
- Failed - The execution of the background task failed. The overall reason “kind” can be seen in the background task filter, along with a more detailed description in the log associated with the task.

In case of a failed task, the Execution Result can have one of the following values:

- Aborted - The execution of the background task was aborted by a user, or automatically by the system because the background task timed out.
- User Error - This happens when the background task attempts to do something that does not make sense or is not allowed--just like an end user can experience an error message when entering data that is not accepted by the system.
- Internal Application Error - This happens in case of an internal error in the application. Situations like this should typically lead to support issues with customer care.
- Data Fail Error - This may happen if the background task targets a record that does not exist, or if the background task declares the creation of an entry that already exists. For example, if you try to update an employee that has been deleted, this kind of error may occur.
- Data Busy Error - This may happen if the data being targeted is in use or being modified by other users while the background task is running. Usually this error will trigger a re-attempt a while later. If this error keeps occurring, the task will fail with this error code at the end.
- Access Error - This may happen if the user executing the background task does not have access to the container in question.
- Setup Error - This may happen, for example, if the task has been set-up to submit a time sheet that is already submitted.
- Other Error - This may happen for other kinds of errors.

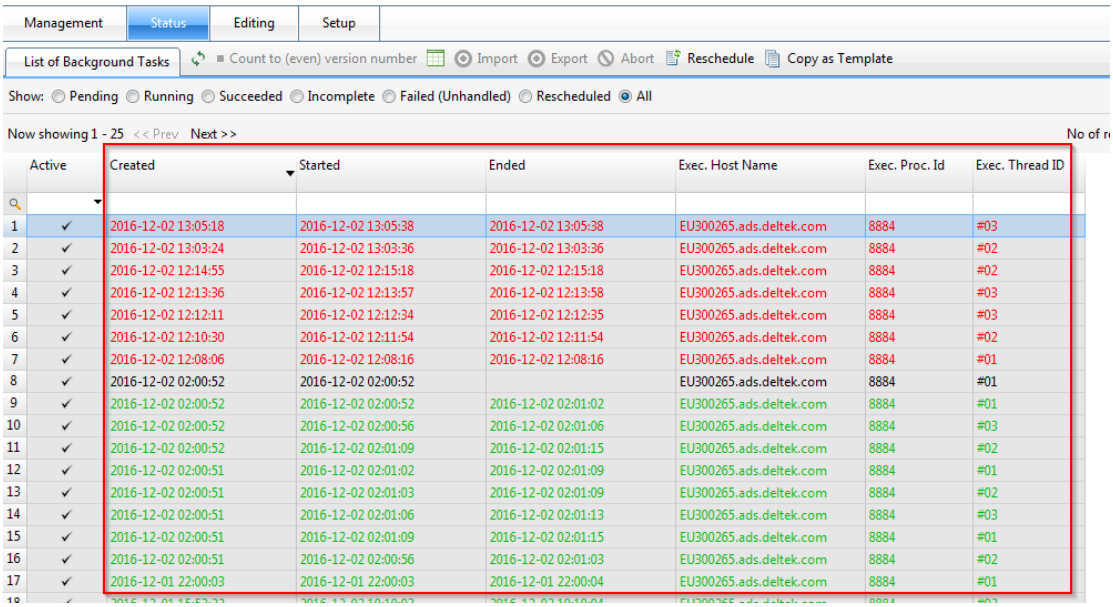
If you go to Set Up / Background Tasks, Status » List of Background Tasks there are many possibilities of overviewing tasks of various categories.



	Id	Task Description	Execution Status	Execution Result	No. Records Processed	Contai
1	Background...		Failed	Setup Error	0	Employ
2	Background...		Failed	Data Fail Error	0	Standa
3	Background...		Failed	User Error	0	TimeSh
4	Background...		Failed	Setup Error	0	TimeSh
5	Background...		Failed	User Error	0	Employ
6	Background...		Failed	Other Error	0	Employ
7	Background...		Failed	Other Error	0	Employ
8	Background...	Generated by RecalcNotificati...	Running		0	UserNo
9	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
10	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
11	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
12	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
13	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
14	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
15	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo
16	Background...	Generated by RecalcNotificati...	Succeeded		1	UserNo

In addition to the actual status, you can trace:

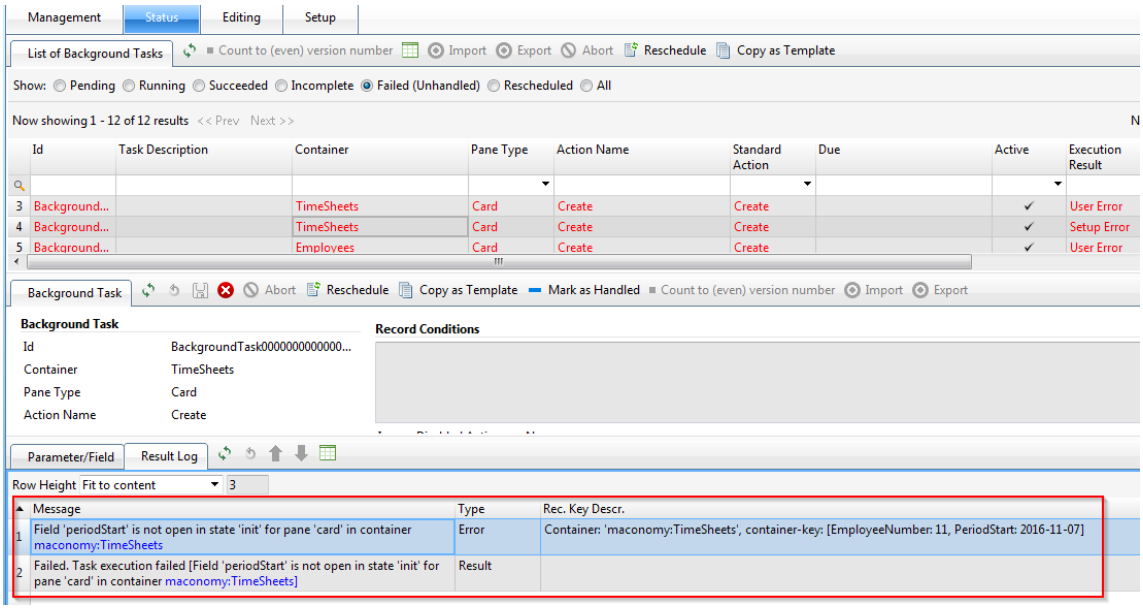
- When was the task created
- When was the task due
- When was the task started
- When was the task ended
- Which host machine / PID / Thread ID executed the task



Why Did a Background Task Fail?

A background task has failed. To understand why, examine the explanations provided in the Log of the background task. The log is found in: Set Up / Background Tasks, Status » List of Background Tasks » Background Task » Result Log.

In the case below, a time sheet was attempted created with the input: EmployeeNumber = '11' and PeriodStart = 2016-11-07, but the field PeriodStart is not open for data entry.



How Can I Limit the Size of the Background Task Database Table?

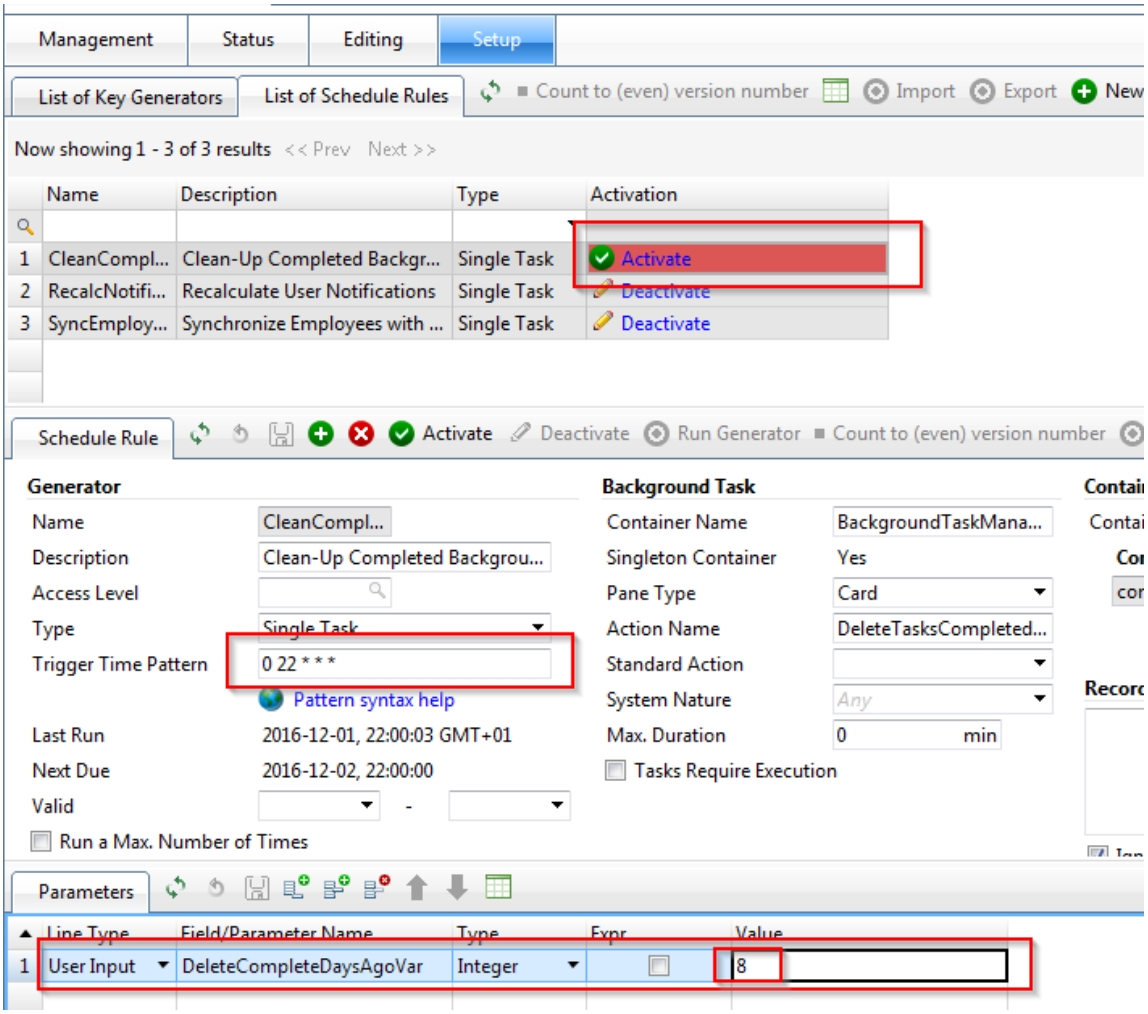
Background tasks make up an important trace of things that has been done, letting a system administrator monitor the status of things.

It's possible to set up a schedule rule that automatically deletes completed background tasks that are older than X number of days. This is even a default rule. By default this rule deletes all completed background tasks that are 8 or more days old.

It is possible to configure:

- Whether this should happen
- What the time range should be
- When the rule triggers

This can be done from: Set Up / Background Tasks, Setup » List of Schedule Rules » Schedule Rules » Parameters.



By editing the rule CleanCompletedBackgroundTasks you can alter this. In this example, if you change the Value field in the table "Parameters" from 8 to 15, tasks will have to be at least 15 days old to be deleted.

Just remember that the longer the duration, the larger the size of the Background Tasks database table.

Can I See Tasks That Were Run as Another User?

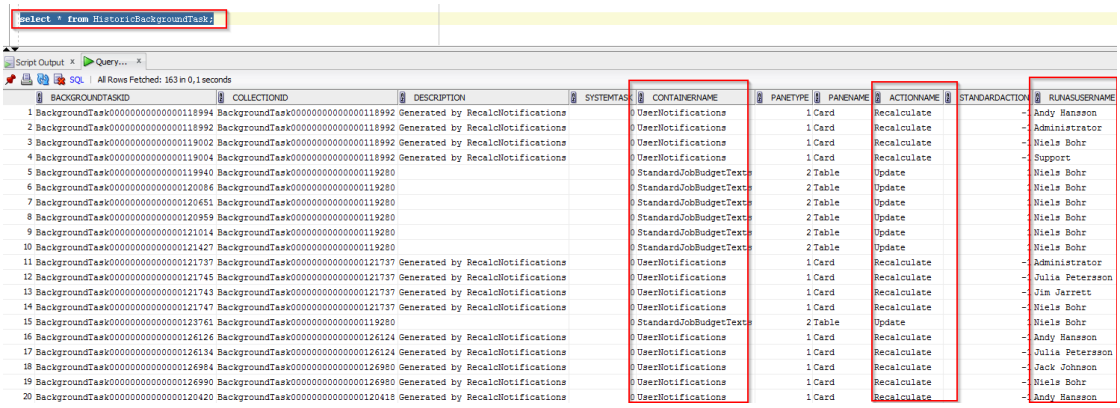
In the background task status filter, you can see whether a background task was run on behalf of another user. If background tasks have been deleted, there may still be a chance that you can dig up some information.

By default any background task that is set-up to run on behalf of some specific user (i.e., not as the background administrator user) is stored in a condensed form in the database table HistoricBackgroundTask. You can choose:

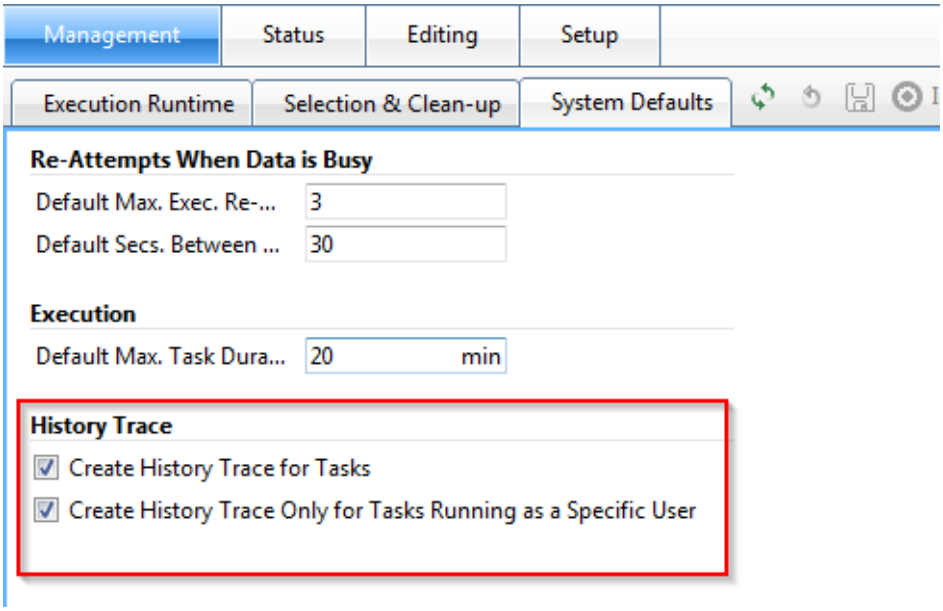
- Whether completed non-failing background tasks are stored in condensed form or not
- If they are, whether only background tasks running on behalf of a specific user are stored like that (in order to keep the size of the table down)

By default, there is no place where you can inspect this historic information in a workspace. And the table is not indexed. This means finding data might take a long time.

Basically, you need to do it using SQLPlus or another database administration tool.



The settings declaring to what extend (if at all) background tasks are logged in the `HistoricBackgroundTask` table, can be altered from Set Up / Background Tasks, Management » System Defaults.



Whenever any of these fields are changed, it is logged (also in HistoricBackgroundTask) who did the change, and what the implication of the change was.

What is the Purpose of the HistoricBackgroundTask table?

The purpose is to resolve a situation where an entity says, for example, “Approved by User name”, but the user refuses any knowledge of that. In this case, we can search for a HistoricBackgroundTask that has done the approval action on behalf of that user. If such a HistoricBackgroundTask exists, the situation can be explained. Obviously this can only be done if history trace is enabled.

How Can I Limit the Size of the HistoricBackgroundTask Table?

You can only limit if/to what extend data is stored in HistoricBackgroundTask. It is not possible to delete any records from this table from within Maconomy. Hence, the table will keep growing as time goes by.

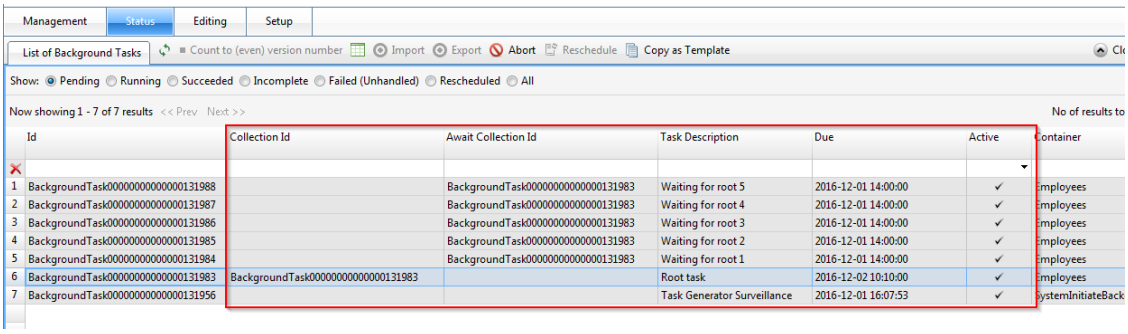
For this reason, it may eventually be necessary to delete “old” portions of this table. Doing so will have to be a manual task done by a DBA, for example using SQLPlus.

Execution Threads are Active; But No Tasks are Being Executed. Why?

You need to figure out whether the background tasks are ready for execution. This includes checking:

- What is the due date/time for the background tasks?
- Are the background tasks active or inactive?
- Are the pending tasks awaiting some other task that isn’t completed, or isn’t due?
- Are the tasks associated a nature different from the nature of the current system?
- Are there errors being reported by the task execution engine (see server log)
- Does the background administrator have adequate access rights?

Properties of the tasks can be seen in: Set Up / Background Tasks, Status » Background Task.



In the above screenshot, you can see that the “Waiting for root n” tasks cannot be picked up for execution because they await the “Root task”. The “Root task” awaits the due date/time before it can be picked up.

Another cause could be related to more critical errors experienced by the background task execution engine. This status of this can be tracked by enabling logging in the `logback.xml` file found in the `configuration` folder in the coupling service installation folder.

Assuming that you have an appropriate “appender” defined (it could be the `FILE` appender, or it could be one dedicated for background tasks execution), you can enable all logging concerning the background task execution engine by adding this to the logback.xml file:

```
<logger name="com.maconomy.coupling.service.batch" additivity="false">
  <level value="DEBUG"/>
```

```
    <appender-ref ref="FILE" />
  </logger>
```

Level TRACE will give an even higher amount of information from the background execution engine. Errors could include, not being able to login, background task user not having adequate access etc.

My Background Tasks Are Not Run at an Adequate Pace – Why?

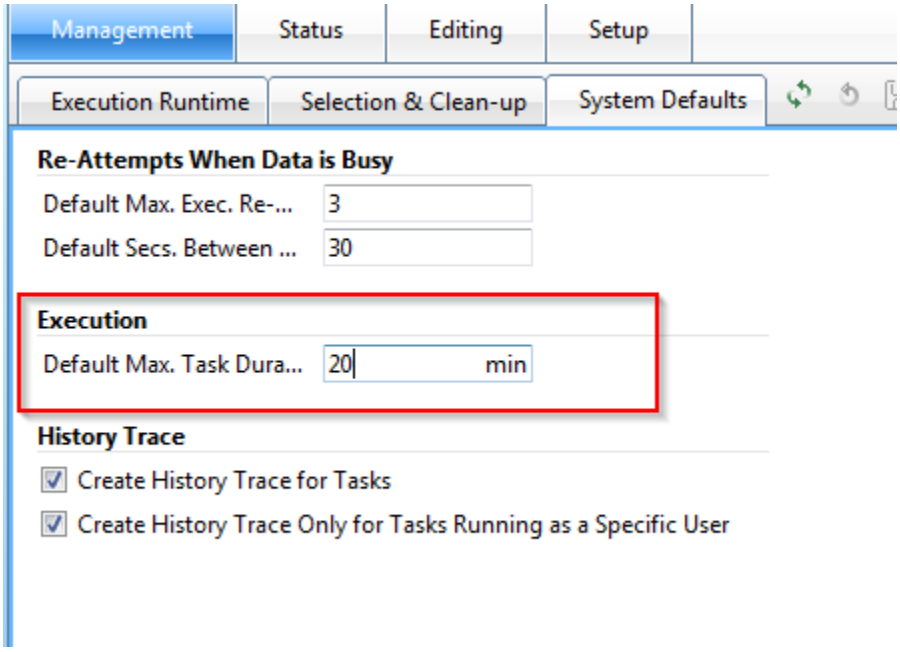
If background tasks are not picked up “soon enough” it may indicate that the execution engine is too busy performing background tasks.

You may need to consider:

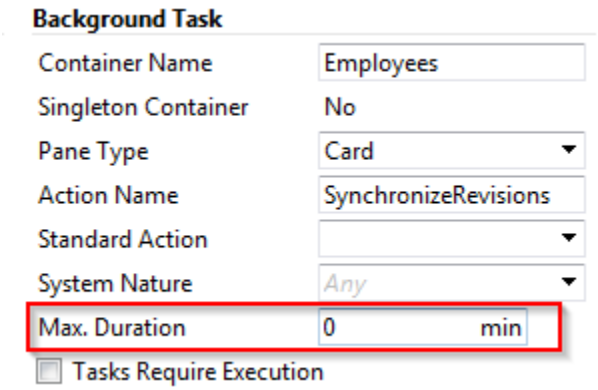
- How many background task execution threads are enabled (across all server machines)
- Are any of the execution threads stopped or suspended?
- Are the pending tasks awaiting other tasks?
- Are the tasks being picked up for execution, but time out? In that case, is the maximum duration of the task set to an adequate amount of time?
- Are any of the expected task assigned a nature different from the current systems?
- Are the tasks activated?

Most of these topic are mentioned elsewhere in this FAQ.

The Maximum duration can be set specifically on a task, as well as on task generator rules. The default maximum duration can be configured in Set Up / Background Tasks, Management » System Defaults.



Maximum duration for tasks created by a schedule rule can be configured in: Set Up / Background Tasks, Setup » List of Schedule Rules » Schedule Rule.



A value of 0 means “apply the default max. duration”.

My Background Task is Green, But It Didn’t Run – Why?

If the task is “green” (i.e., “Succeeded”) it means that the execution framework believes (and has no reason not to) that the task was successfully run.

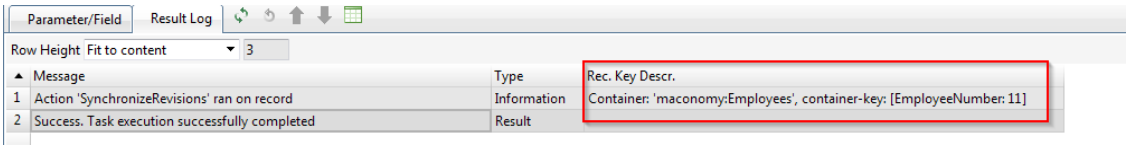
There are a couple of things you can check:

- Is the record key(s) referencing the expected record(s)? If not, the action was executed on something other than what you expected.
- Is the record condition (if any) ruling out the expected record(s)? If so, the records you thought was being processes has been skipped.
- How many records have been executed? If no records was actually executed, this is by default considered a successful background task. It is possible to flag a background task so that if the specified action is not applied for any record, it will be considered an error.

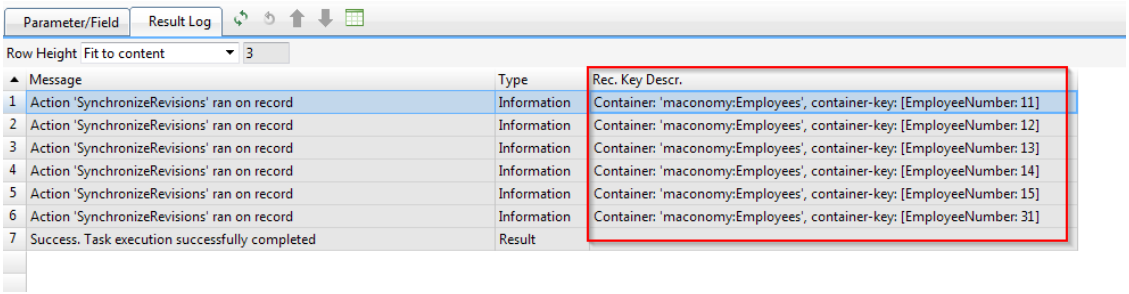
Notice:

In the log for the background task (in the Workspace) you can see exactly which record keys the action has been applied to! Go to: Set Up / Background Tasks, Status » Background Task » Result Log.

In this case, the action ran on one record: (EmployeeNumber = 11)



In this case, the action ran on several records: (EmployeeNumber in {11, 12, 13, 14, 15, 31})



My Background Tasks Generator Hasn't Run - Why?

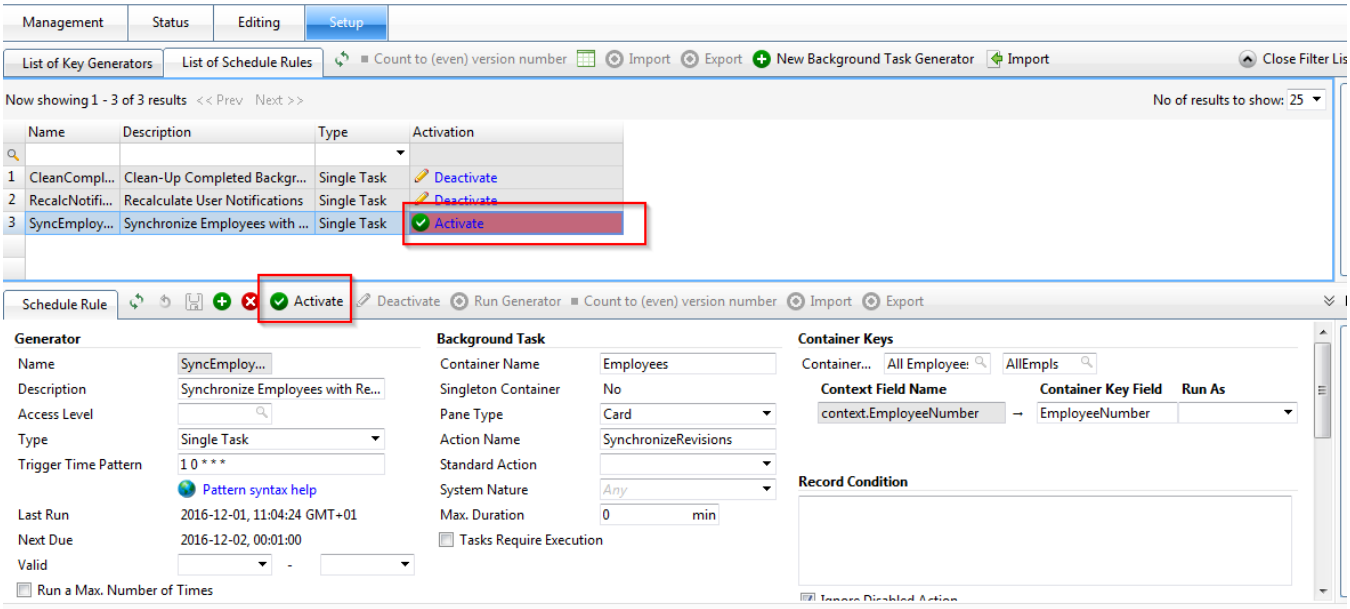
You should investigate:

- When is the next due date/time of the background task generator? If this is sometime in the future, maybe the time trigger pattern is not what you think, and the rule really shouldn't have run
- When was the task generator last run?
- What is the validity date interval for this rule? If the rule is no longer valid (or not yet valid) this explains why.
- Has the rule been set up to run a fixed number of times? If so, are there any runs left? If not, this could explain why.
- Is the rule active? If not, this may explain why.

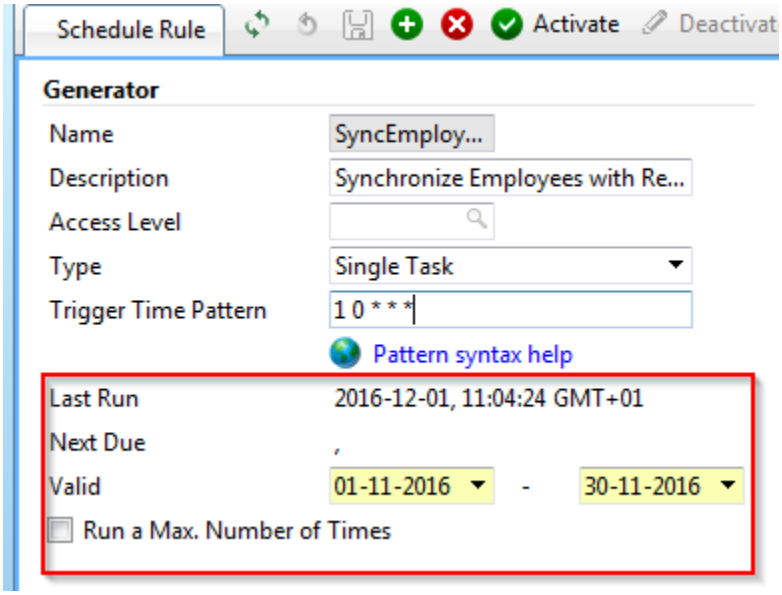
Notice:

You can investigate these things by going to: Set Up / Background Tasks, Setup » List of Schedule Rules » Schedule Rule.

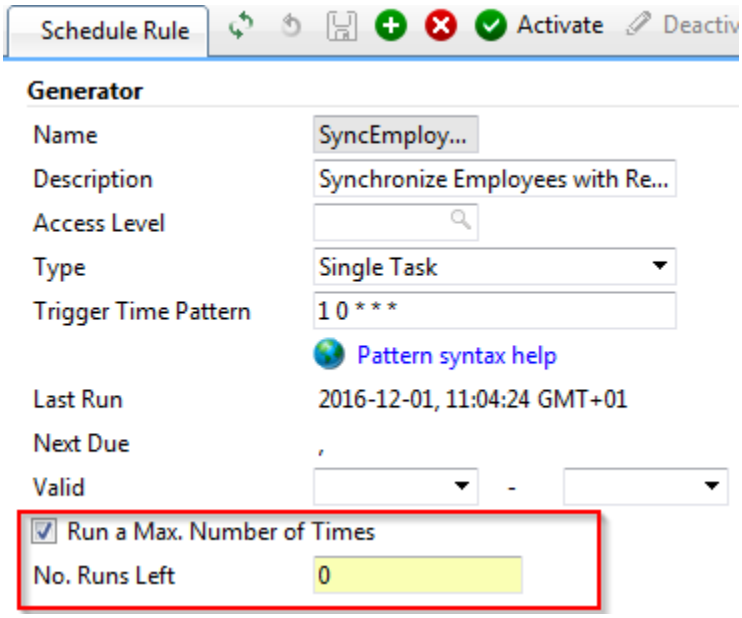
If you can activate the rule, it is currently inactive.



Other settings can be seen here:



If the rule is set to be run a maximum number of times, check if there are no runs left:



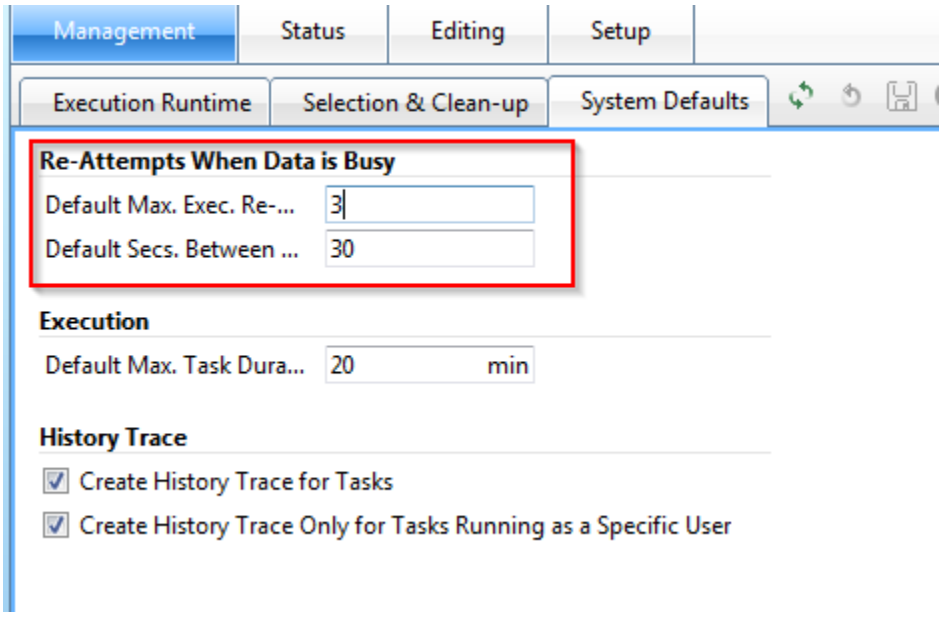
What Happens If Other Users are Changing the Data of a Background Task?

Just as “normal” users can experience that someone else changes that the user is working with, this can happen for background tasks. In case of a “Data has been changed by another user”, the execution engine will automatically detect this situation and will re-schedule the background task to be attempted again “soon”. The number of times this “retry” will occur can be configured, as can the interval between retries. If the task keeps failing for this reason, it will eventually be marked as failed with a “Data Busy” error indication.

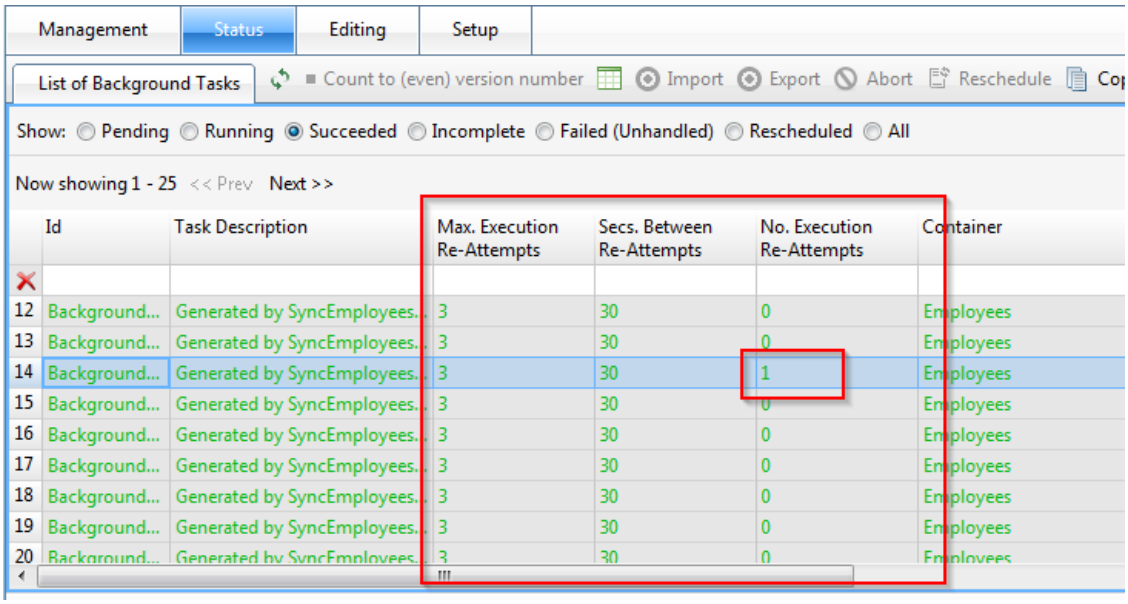
You can track how many times a given task has been re-attempted.

To set-up the default number of re-tries and duration between re-tries, go to Set Up / Background Tasks, Management » System Defaults.

Here you can specify the default number of re-tries and the time between re-tries.



It is possible to look-up what the settings for a given task is, including the number re-tries that has occurred.



Where Did the Output of a Background Task End Up?

When a background task executes, sometimes output documents need to be taken care of. This could be, for example, printed journals, invoices or log files (e.g., from import programs).

If a background task produces output, it needs to be handled. If it is not specified how to handle output files (but the “core logic” otherwise succeeded) the task ends up having status “Incomplete”.

The background task log will show what output documents were produced and how/if they were handled.

Basically, output files can be handled in two ways:

- Sent by e-mail to a specific recipient

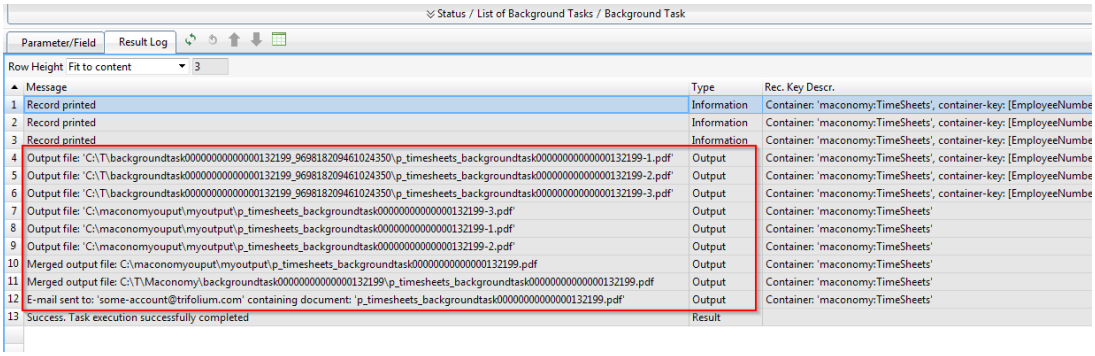
- Stored on the file system somewhere

You can associate a file handler to a “pattern”. If you do, only output files having a name that matches will be handled by that handler. In this way, you can have PDF-files stored in one location, and TXT-files stored in another location.

You can also choose to have some files both sent by e-mail and stored on a file system.

To see what output documents were produced and how they are handled, go to: Set Up / Background Tasks, Status » List of Background Tasks » Background Task » Result Log.

In the example below, files where stored on a file system and sent by e-mail.



Data Import Packages Troubleshooting

There are various means that you can troubleshoot Data Import Packages, including analyzing checksums, package integrity, and package history.

Analyze Checksums

Checksums show unique identifiers of this package, used for troubleshooting by advanced users. You can track the package throughout the system based on these checksum identifiers. These checksum fields can be used to verify the integrity of the package and track the package and package lines throughout the different systems.

When you create a data import package, it is assigned a unique identification called Base Checksum. All the package lines created in this package are assigned the same Base Checksum. This checksum is assigned once and it never changes. This checksum can be used to track the package and package lines throughout different systems, as well as validate the origin.

Loaded Checksum and Saved Checksum fields are used when loading or saving packages from or to a file. When a package or package line is loaded or saved, then the Current Checksum is transferred to the loaded/saved checksum. In this way it is possible to track if the package has changed since it was loaded from a file or saved to a file.

Additionally, the Imported Checksum field on package lines is set once the line is successfully imported. Maconomy transfers the value of the Current Checksum to this field. The Imported Checksum on the package holds the accumulated checksum of all the imported lines. This means that if all the lines in the package have been successfully imported then the value in this field will be equal to the value of the Current Checksum field.

To analyze checksums:

1. Go to **Setup » Data Import Packages » Data Import Package tab » Advanced sliding panel**.

2. Review the checksums as needed and compare the unique identifiers as described above.

Note: Additionally, an Advanced sliding panel for checksums is available in the sub-tab.

Verifications and Integrity Checks

The Data Import Package workspace offers numerous fields that help you verify import data package information necessary for troubleshooting and confirming data integrity.

To perform verification and integrity checks:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. In the Integrity island, review the **Changed After Load** and **Changed After Save** fields.

   A red color indicates a potential integrity breach.
3. In the Package Information island, review validation and import details, including wall clock duration and execution duration.
4. In the Last Execution island, review who scheduled last validations and imports, as well as when the action was performed.

Package History

A record of import data package information is retained for troubleshooting and data review purposes to track the lifetime of a package from its inception to import and purge. Details to help troubleshoot include the type of change, such as status change or import, the user who made the change, and the date.

The information is retained as:

- Summary format on the History tab.
- Detailed format on the All Lines History tab.

To review package history:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. Review data on the **History** or **All Lines History** sliding panel.

Cancel Validation or Import

Sometimes an issue occurs and you must cancel a validation or import.

To cancel a validation or import:

1. Go to **Setup » Data Import Packages » Data Import Package tab**.
2. Click **Cancel Validation** or **Cancel Import**, depending on which mode you are in.
3. A dialog displays to confirm. Click **OK**.

After cancellation, correct the error and reschedule the validation or import.

Note: This information is logged in the Last Execution island on the Data Import Package tab.

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.

Maconomy Server Executable 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.

Index for MACONOMY_USERS

To improve the index for MACONOMY_USERS, the command lines are migrated from the server command line i/f to the CouplingService command line i/f.

For example, using the `w_21_0.sp103` installation with shortname `w21p3`, a re-import and compile of print layouts looklike the following:

```
Cd C:\Maconomy\w_21_0. sp103\CouplingService
CouplingService.exe -S m21p3 -UP
```

Notes:

- The process must occur in the folder where CouplingService.exe is installed.
- A space MUST follow the '-S' option.

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.
--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.
-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.

Option	Description
-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. **Warning:** 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: 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

Option	Description
	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

--CompileAnalyzer <path>	With --CompileAnalyzer, the server will compile Analyzer reports. Use the following command:
	
	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.

Option	Description
--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.
--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`

Option	Description

	`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 `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\GeneralMonitors`. See the Maconomy Workflow Language Reference manual for details.

Option	Description
--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.
--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.
	
	`--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.

Option	Description
	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:

Option	Description
	<x>Debug<y..y>, where <x> can have the following values: O When running an Oracle™ 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 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
`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	

Name	Type	Description
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, 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 when a report is printed. Activate the caching system to improve performance. If the value is false, the other RO* options do not apply.
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.
RPC_UserIdleTimeout	INTEGER	The timeout of idle users connected via the RPC protocol. This includes the Workspace Client and REST API.

Name	Type	Description
`RPC_Log = true|false`	BOOLEAN	The collected information will 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	Comma-separated list with log events on which to truncate the RPC log. Any combination of the following log events can be set: login, reconnect, detach, logout, all
`DB_DisableRequestCancellation`	BOOLEAN	This flag disables progress monitoring and cancellation for long running database operations. The default value is 'false'.
`DB_Cancellation_InitialDelay`	INTEGER	The initial delay in seconds before the progress bar appears for long running database operations. The default is 2 seconds.
`DB_Cancellation_PollInterval`	INTEGER	The maximum reaction time in seconds for the progress bar if the user decides to cancel the long running database operation. The default is 2 seconds.
`DB_Cancellation_DefaultTimeout`	INTEGER	The default timeout before a long running database operation is automatically cancelled. MConfig will set this value, but otherwise the internal default is 300 seconds (5 minutes).
`DB_CancellationCompletionTimeout`	INTEGER	The maximum time the server will wait for the database to respond to a cancellation request. If the running database request is not successfully cancelled within this time the database session will be forcefully terminated. The default value is 10 seconds.

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.
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”.

Name	Type	Description
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 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.

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
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)

Digit	Pertains to	Function	Default
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>*`

Name of Executable	Description
	`-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.”
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.

Name of Executable	Description
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.
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

Name of Executable	Description
	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.
vttool.exe	For internal use.

Script Files

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

> **Note:** 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.
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.

Name of Command File	Description
UpdateFieldsAndRelations.w_8_0.cmd	Used in connection with upgrades.

Coupling Service 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> — This port is used for Maconomy 2.0 web services and by the Maconomy 2.0 Workspace Client to connect to the Coupling Service. It must be accessible from all network locations where users are expected to connect.

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.proxy.encryption=<true/false> — Specifies whether proxy encryption has been configured between the Coupling Service and external clients.
- 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.

> **Note: 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.

Displaying System Information on Clients

Companies may want to display the details of the system to which their Maconomy installations are connected. While system information is not displayed by default on any client, you can configure your Workspace and web clients to display this information on their respective interfaces. When you perform this configuration, the system information is displayed on the menu dock in the Workspace Client (that is on the left side/margin of the interface), and along the top margin in the corresponding web client.

To display the system name on both your Workspace and web clients, configure the content of the menu dock in the MCSL file of the Coupling Service. The configuration might look as follows:

```
<Binding namespace="client:menudock">
   <Fields>
      <Field name="Title" valueString="^{'Logged into: ' + shortname()}"/>
   </Fields>
</Binding>
```

Additional Specifications for the Workspace Client

Other attributes you can specify to customize how the system name is displayed in the Workspace Client are:

- Color – To specify a color, assign a “Color” field to a valueString that uses one of the following 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)
- TopToBottom, which indicates whether the text is drawn top-to-bottom (this is the default) or bottom-to-top (if you specify the value “false”).
- FontHeight, which indicates the font height to use. (The default is 12.)
-
- Your configuration might look like the following example:

```
<Binding namespace="client:menudock">
   <Fields>
      <Field name="Color" valueString="rgb=#abba76" />
      <Field name="TopToBottom" valueString="false" />
      <Field name="FontHeight" valueString="8" />
   </Fields>
```

- </Binding>
-

Additional Specifications for Web Client

If you want to specify background and foreground color-coding for your corresponding web client installation, immediately after the "client:menudock" specification, you can enter something like the following example:

```
<Binding namespace="client:menudock:web">
```

```
   <Fields>
      <Field name="backgroundcolor" valueString="#ff0000"/>
      <Field name="foregroundcolor" valueString="#ffffff"/>
   </Fields>
</Binding>
```

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

Transcript of a ssh-host-config Session

Here is a complete transcript of an `ssh-host-config` session:

```
$ ssh-host-config

*** Info: Generating missing SSH host keys
ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
*** Info: Creating default /etc/ssh_config file
*** Info: Creating default /etc/sshd_config file

*** Info: StrictModes is set to 1yes1 by default.
*** Info: This is the recommended setting, but it requires that the POSIX
*** Info: permissions of the user1s home directory, the user1s .ssh
*** Info: directory, and the user1s ssh key files are tight so that
*** Info: only the user has write permissions.
*** Info: On the other hand, StrictModes don1t work well with default
*** Info: Windows permissions of a home directory mounted with the
*** Info: 1noacl1 option, and they don1t work at all if the home
*** Info: directory is on a FAT or FAT32 partition.
*** Query: Should StrictModes be used? (yes/no) yes

*** Info: Privilege separation is set to 1sandbox1 by default since
*** Info: OpenSSH 6.1.  This is unsupported by Cygwin and has to be set
*** Info: to 1yes1 or 1no1.
*** Info: However, using privilege separation requires a non-privileged account
*** Info: called 1sshd1.
*** Info: For more info on privilege separation read
/usr/share/doc/openssh/README.privsep.
*** Query: Should privilege separation be used? (yes/no) yes
*** Info: Note that creating a new user requires that the current account have
*** Info: Administrator privileges.  Should this script attempt to create a
*** Query: new local account 1sshd1? (yes/no) yes
*** Info: Updating /etc/sshd_config file

*** Query: Do you want to install sshd as a service?
*** Query: (Say "no" if it is already installed as a service) (yes/no) yes
*** Query: Enter the value of CYGWIN for the daemon: []
*** Info: On Windows Server 2003, Windows Vista, and above, the
```

```
*** Info: SYSTEM account cannot setuid to other users -- a capability
*** Info: sshd requires.   You need to have or to create a privileged
*** Info: account.   This script will help you do so.

*** Info: It1s not possible to use the LocalSystem account for services
*** Info: that can change the user id without an explicit password
*** Info: (such as passwordless logins [e.g. public key authentication]
*** Info: via sshd) when having to create the user token from scratch.
*** Info: For more information on this requirement, see
***  Info:  https://cygwin.com/cygwin-ug-net/ntsec.html#ntsec-nopasswd1

*** Info: If you want to enable that functionality, it1s required to create
*** Info: a new account with special privileges (unless such an account
*** Info: already exists). This account is then used to run these special
*** Info: servers.

*** Info: Note that creating a new user requires that the current account
*** Info: have Administrator privileges itself.

*** Info: No privileged account could be found.

*** Info: This script plans to use 1cyg_server1.
*** Info: 1cyg_server1 will only be used by registered services.
*** Query: Do you want to use a different name? (yes/no) no
*** Query: Create new privileged user account 1NODE\\cyg_server1 (Cygwin name:
1cyg_server1)? (yes/no)
*** Info: Please enter a password for new user cyg_server.   Please be sure
*** Info: that this password matches the password rules given on your system.
*** Info: Entering no password will exit the configuration.
*** Query: Please enter the password:
*** Query: Reenter:

*** Info: User 1cyg_server1 has been created with password 1<something>1.
*** Info: If you change the password, please remember also to change the
*** Info: password for the installed services which use (or will soon use)
*** Info: the 1cyg_server1 account.
```

```
*** Info: The sshd service has been installed under the 1cyg_server1
*** Info: account.  To start the service now, call 'net start sshd1 or
*** Info: 'cygrunsrv -S sshd1.  Otherwise, it will start automatically
*** Info: after the next reboot.

*** Info: Host configuration finished. Have fun!
```

COMMANDS

Copy to come.

Appendix

Analyzer

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.

Attribute Name	Type	Usage
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

Attribute Name	Type	Usage
		• 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.

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).

A Maconomy system contains several different server components and different client interfaces.

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.

About Deltek

Better software means better projects. Deltek is the leading global provider of enterprise software and information solutions for project-based businesses. More than 23,000 organizations and millions of users in over 80 countries around the world rely on Deltek for superior levels of project intelligence, management and collaboration. Our industry-focused expertise powers project success by helping firms achieve performance that maximizes productivity and revenue. www.deltek.com