ARIS Process Performance Manager

SYSTEM ARCHITECTURE

Version 10.2

April 2018

This document applies to PPM Version 10.2 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Contents

1 Text conventions

Menu items, file names, etc. are indicated in texts as follows:

- Menu items, key combinations, dialogs, file names, entries, etc. are displayed in **bold**.
- User-defined entries are shown in **<bold and in angle brackets>**.
- Single-line example texts (for example, a long directory path that covers several lines) are separated by ↵ at the end of the line.
- File extracts are shown in this font format:

  ```
  This paragraph contains a file extract.
  ```

2 General

The manual describes the architecture and operation of ARIS Process Performance Manager, or just PPM. It explains the relationships and data flows between the PPM system components and introduces various installation types. In addition, the architecture and operation of the software as well as the installation settings and their effects are documented.

It provides the technical administrator of the PPM system with basic knowledge and configuration expertise to help him to implement, adapt, and maintain different system environments.

The manual is not intended to replace user or customizing training. It is rather a reference containing supplementary information concerning other PPM manuals and the PPM online help.

3 Overview

ARIS Process Performance Manager is a tool designed to analyze actual working processes. The data comprising the steps of these processes is extracted from application systems and consolidated to process instances. Then, measures are calculated for these process instances. In this way, PPM provides a comprehensive overview of the processes in a company and supports the user in identifying weak spots.

PPM uses an RDBMS as the repository in which all configurations and data are saved. PPM has been developed in Java as a client-server application.

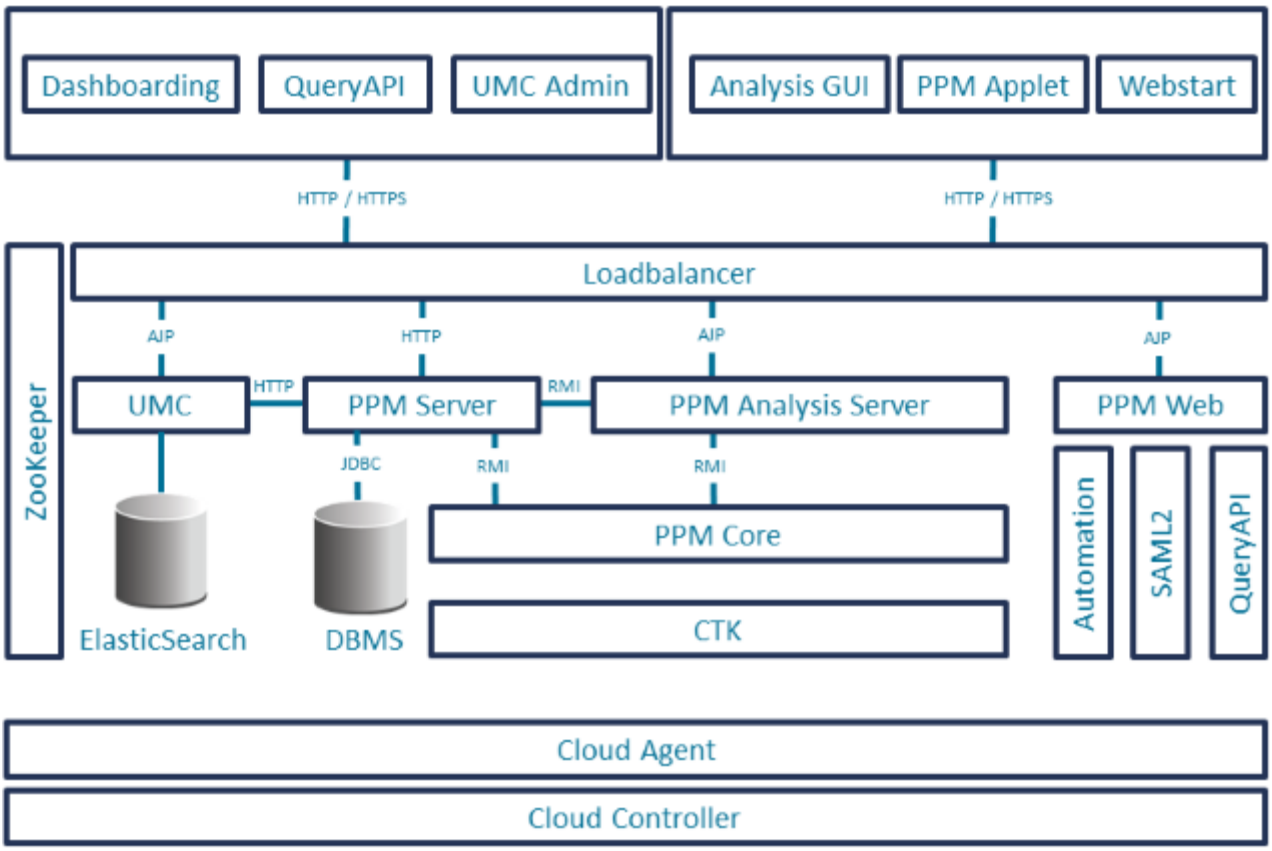
3.1 The PPM client concept

The PPM client concept facilitates independent management of different data sets within a PPM installation. An independent database schema is assigned to each PPM client. When logging in to the PPM system, the user specifies the required client. The database schemas can be distributed to several database servers.

PPM clients are created interactively using PPM Customizing Toolkit (or just CTK). Detailed information on this topic is available in the **PPM Customizing Toolkit** manual.

3.2 System components

The core components of PPM are the PPM server and the PPM analysis server. Their function is based on additional infrastructure components. PPM functions are accessed via a graphical user interface. The following image represents an overview of all PPM system components.



3.2.1 Infrastructure

The infrastructure consists of these components:

- Cloud Agent,
- Cloud Controller,
- Apache ZooKeeper,
- Apache Load Balancer,
- Elasticsearch
- Central user management.

During installation, the user selects the memory model (S, M, or L) to be used for the services included. The service name contains the selected memory model as a suffix.

3.2.1.1 Cloud Agent

Cloud Agent is a service that enables you to install, configure, start, stop, and monitor product and infrastructure components. Cloud Agent is set up during installation and starts automatically. Under Windows, Cloud Agent is set up as a service called **Software AG PPM <version>**. Under Linux, Cloud Agent can be set up after installation as a daemon process. You need root privileges to do so, though.

3.2.1.2 Cloud Controller

Cloud Controller is a command line program used for sending commands to an active Cloud Agent.

3.2.1.3 Apache Load Balancer

The load balancer is connected upstream of the Web application servers and distributes the incoming queries to the relevant Web servers and applications.

Apache Load Balancer is set up in Cloud Agent as an infrastructure component named **loadbalancer_<memorymodel>**.

3.2.1.4 Apache ZooKeeper

Product and basic infrastructure components use Apache ZooKeeper as a central registration service. Apache ZooKeeper centrally saves the configuration information and names of the registered components. It enables distributed synchronization and creates group services.

Apache ZooKeeper is set up in Cloud Agent as an infrastructure component named **zoo_<memorymodel>**.

3.2.1.5 Elasticsearch

Elasticsearch is a real-time search and analysis application. Within the architecture described, it is used for saving user management and revision data.

Elasticsearch is set up in Cloud Agent as an infrastructure component named **elastic_<memorymodel>**.

3.2.1.6 Central user management (User management)

The Web-based central **User management** manages users, user groups, and product licenses. User data can be managed in the **User management** component by users with the **Administrator** function privilege. To register PPM, the product license must be imported into central user management after installation.

Central user management is set up in Cloud Agent as an infrastructure component named **umcadmin_<memorymodel>**.

3.2.2 PPM components

3.2.2.1 PPM user interfaces

PPM applet, analysis GUI, and query API are the PPM front-end. With the PPM front-end, you can analyze and visualize previously calculated process characteristics.

3.2.2.2 PPM client server

Each PPM client server consists of a PPM server and an associated analysis server. Both servers are implemented as independent services and can be started and stopped individually. The PPM server is an interface for the PPM front-end and coordinates access to the analysis and database server.

The analysis server contains all process data for efficient analyses in an in-memory structure (see Basic information on in-memory technology (Page 7)).

PPM server and analysis server are set up in Cloud Agent as product components named **<client>_cs** and **<client>_as**.

3.2.2.3 PPM Web

PPM Web is an application server offering the following services:

- SAML2 log-in
- Automation
- Query API

As a Web server, PPM Web also supplies the PPM applet.

PPM Web is set up in Cloud Agent as a product component named **PPM_web**.

External access to the services provided by PPM Web is executed via load balancer (see Apache load balancer (Page 5))

3.2.2.4 PPM Core

PPM Core consists of RMI and CORBA registry. These registries are name services that the servers use for registering their services.

The PPM server components (analysis server and command line programs) use the RMI registry to query the addresses of the PPM servers.

PPM Core is set up in Cloud Agent as a product component named **PPM_core**.

3.2.2.5 Customizing Tool Kit (CTK)

CTK manages clients and their configurations. In addition to system configuration including language settings, memory settings, data sources, and internal access data, it is possible to define custom job automations and edit the process types, measures, dimensions, and attributes available in the analysis.

3.2.3 Database server (DBMS)

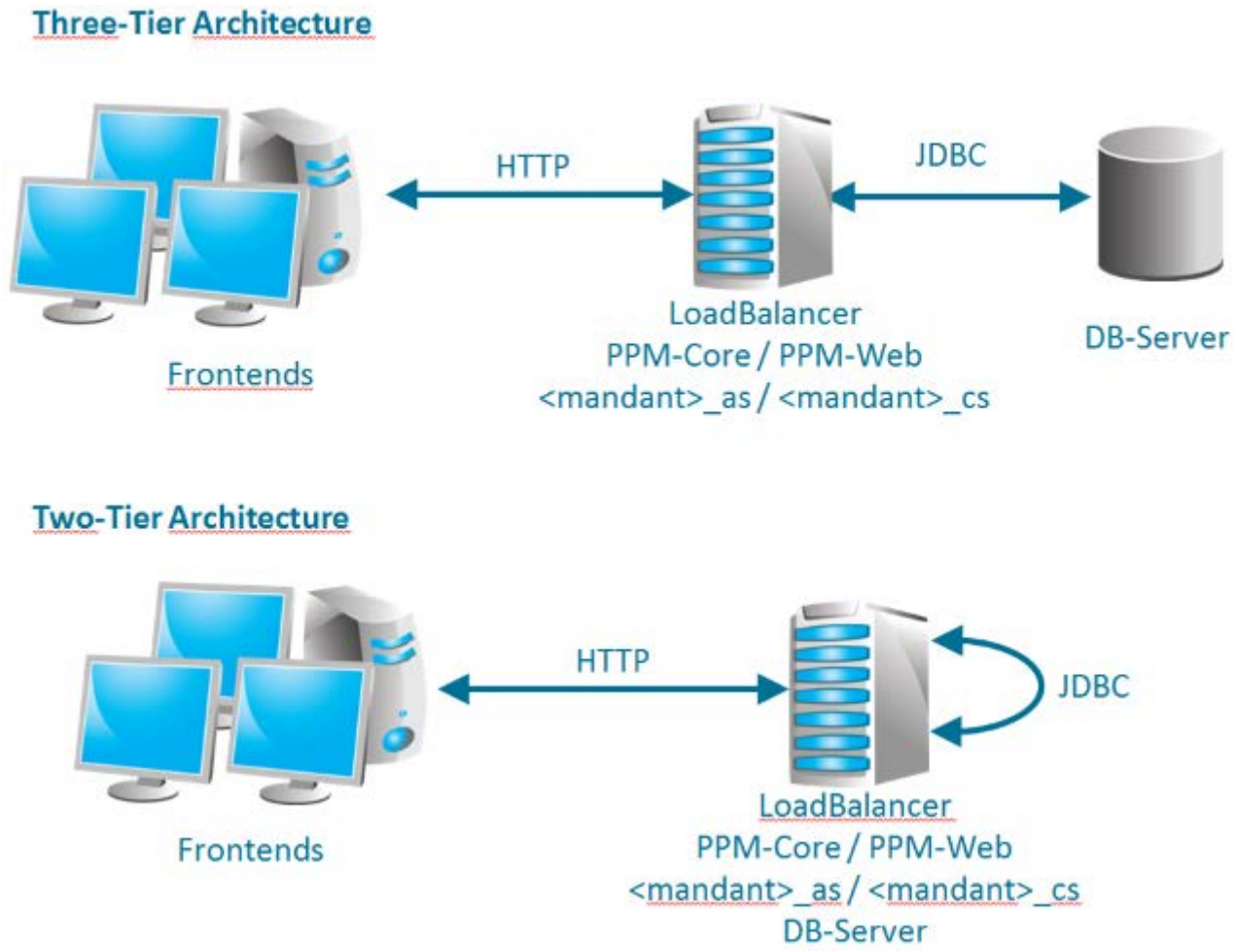
The database server (DBMS) is a persistence layer for the PPM server and data import. In addition to configuration, administration, and user data (such as favorites), all process instances and Data

analytics data are stored here. In case of system failure, they can be used for restoring the analysis server.

For each client, a database schema to be created in the DBMS is used. Saving cross-client data is not required. This means that different database instances can be used for different clients.

Depending on the system architecture required, the database system can be installed on the PPM server itself (two-tier architecture) or on a server available in the network. Since large data volumes are exchanged especially during data import, a fast network connection between the servers should be ensured when using an external database (three-tier architecture).

Detailed information is available in the **PPM Database systems** manual.



3.3 Basic information on in-memory technology

Due to the PPM in-memory technology, analysis data is managed in a compact main memory structure optimized for analysis queries. This enables the system to respond to queries a lot quicker than with conventional, hard drive based technologies.

The analysis server manages the main memory structures. To facilitate memory management and enable individual reboots of the PPM server, the server is started in a separate process. Both processes must be active in a client for it to be able to function.

During data import, the special data structures of the analysis server are filled with all data required for the analysis. When new data is imported into the PPM system, the analyses structures are automatically updated or new structures are created, if required.

To keep downtime to a minimum after terminating the analysis server (for example, after a PC reboot or system error), the analysis server creates a recovery file. The analysis server uses this file to quickly restore itself completely after a reboot. The initial recovery files are automatically created during system initialization and recreated in a configurable way (default: activated) during each data import. It is also possible to trigger regeneration via the administration command line program **runppmadmin**. This ensures that changes to the configuration are also updated in the recovery files, if required.

Obsolete or missing recovery files are automatically identified by the system. In these cases, the analysis server and associated recovery files can be newly generated from the database.

4 PPM network

PPM uses the default protocols of the TCP/IP network to exchange data. This chapter provides a brief overview of the exchange of data within a computer network, the data flows between the PPM components, and possible PPM implementation scenarios.

4.1 Network protocols

GENERAL

Information exchange on the Internet is based on the TCP/IP network protocol (Transmission Control Protocol/Internet Protocol) and executed by means of direct socket connections. The TCP/IP protocol is a fault-tolerant protocol that detects and eliminates transmission errors.

The computer's IP address and a port number uniquely specify a socket. The server program generates a socket upon boot-up and waits for requests. A client computer connects to the server by generating a socket and connecting it to the socket of the server. Next, the information exchange takes place within the protocol set by the server socket. The connection is terminated when one of the partners closes the socket. This is usually the client because the server waits for further requests.

The port numbers up to 1023 are reserved for specific transmission protocols. They are called well-known ports and must only be used for the respective protocols. For example, the following fixed server port numbers (in parentheses) have been assigned on the Internet to particular transmission protocols: HTTP (80), FTP (21), telnet (23), SMTP (25), POP3 (110), and HTTPS (443). Port numbers above 1023 are called user ports.

In contrast to the default port numbers, you can choose the port number for most additional services.

A socket connection within a network must be unique. A port can only be assigned once for each network address.

NETWORK INTERFACES

Usually, each server or desktop PC on which PPM can be installed has one to two network resources or cards (also called NIC = network interface card). Each NIC provides a unique address (IP) within the network at which the server can be reached. Usually, one NIC is sufficient for operating PPM. Each service provided by PPM runs on an individual IP and port combination unique for the NIC.

If multiple NICs that can be used by an application (management networks excluded) are available in the server, PPM can also be configured on individual NIC addresses (IP + port). Using multiple network cards becomes interesting if the number of available ports is limited.

PPM NETWORK PROTOCOLS

Communication among the PPM components is based on TCP/IP. Other protocol families (for example, IPX, SPX) are not supported. PPM uses the following TCP/IP protocols:

Component	Protocol	Default port
Load balancer	HTTP/HTTPS	4080/4443
CORBA registry	Activation port	17589
RMI registry	RMI/SSL RMI/zipped RMI	17500
<client name>_cs	RMI/SSL RMI/zipped RMI	17501 + 2*n
<client name>_cs	CORBA (IIOP)	17590 + n
<client name>_cs	HTTP/HTTPS	17651 + n
<client name>_as	RMI/SSL RMI/zipped RMI	17502 + 2*n
PPM Web	HTTP	17101
PPM Web	AJP	17201
Elasticsearch	HTTP	17047
ZooKeeper	HTTP	17050
UMC	HTTP	17100
UMC	AJP	17200
Cloud Agent	HTTP/HTTPS	17004

RMI (REMOTE METHOD INVOCATION)

The PPM server components use this protocol to exchange data. PPM uses the protocol variant **JRMP** (Java RMI Message Protocol) as the transfer standard. By default, the RMI protocol requires a direct socket connection.

CORBA (COMMON OBJECT REQUEST BROKER ARCHITECTURE)

The query API uses this protocol to communicate with the PPM client server within a server installation.

JDBC (JAVA DATABASE CONNECTIVITY)

Java applications use this standardized protocol to access an RDBMS. The JDBC drivers required for this are not part of the PPM installation and are provided by the database manufacturer.

HTTP (HYPER TEXT TRANSFER PROTOCOL)

The HTTP protocol is used for all data exchange between the PPM front-end and the PPM server.

PPM uses this protocol only to transfer the HTML document pages, text-based configuration files (*.properties), and Java archive files required to run the PPM user interface in the browser.

The Web application server also uses this protocol to publish Management views and to run Performance Dashboard.

Instead of the HTTP protocol, you can also use the secure variant, **HTTPS**.

4.2 Loading the PPM front-end

The PPM front-end can be loaded and run in the following three different modes.

AS A JAVA APPLICATION IN A COMMAND LINE ON THE SERVER

You can start the PPM front-end directly on the server, in the installation environment. To do this, open a command line on your computer and go to the following directory:

<PPM installation directory>/ppm/server/bin/agentLocalRepo/.unpacked/<installation_time>_ppm-client-run-prod-<PPM Version Info>-runnable.zip/ppm/bin

and **runppmgui.bat** for Windows operating systems or **./runppmgui** for Linux to start the PPM front-end.

AS A JAVA APPLET IN A WEB BROWSER ON THE CLIENT

Depending on user interaction, JAR files are transferred from the PPM Web server to the PPM client and executed in the Java Runtime Environment of the browser plug-in. To start the PPM front-end in the Web browser, enter the following URL in the browser's address bar:

http://<loadbalancer>:<loadbalancer-http-port>/ppm/html

Example

http://localhost:4080/ppm/html

AS A JAVA WEBSTART APPLICATION ON THE CLIENT

Depending on user interaction, a JNLP file is downloaded from the PPM Web server to start the WebStart application, and this is then executed in the client's Java Runtime Environment. In some Web browsers, the PPM front-end is started automatically by entering the following URL in the browser's address bar:

http://<loadbalancer>:<loadbalancer-http-port>/ppm/html/ppm.jnlp

Example

http://localhost:4080/ppm/html/ppm.jnlp

If you are using the Chrome Web browser, the **ppm.jnlp** file is first downloaded and must be run locally. In both cases, however, the PPM front-end is started as an application on the client.

4.2.1 HTTPS protocol

If you want to use the secure HTTPS protocol instead of the HTTP protocol, you only need to change the URL for starting the PPM front-end as follows.

https://<loadbalancer>:<loadbalancer-https-port>/ppm/html

Example

https://localhost:4443/ppm/html

The HTTPS protocol is already activated in load balancer and can be used without changing the configuration. By default, a signed certificate generated by the load balancer is used for HTTPS

support. Most browsers do not support it, however, because it does not contain any valid root certificate of an approved certification authority. The **PPM Operation Guide** describes how to create a valid certificate for HTTPS support of the load balancer.

Using the HTTPS protocol for the load balancer secures all communication between the PPM front-end and the PPM server. The **PPM Operation Guide** describes how to encrypt RMI data communication between the PPM server components.

4.3 Data communication between PPM front-end and PPM client server

All data exchange between the PPM server and PPM front-end is based on SOAP web services. The PPM user interface calls methods in its environment that are executed on the PPM server. The functions of the PPM server are realized in SOAP web service classes.

The PPM front-end logs into the load balancer using the PPM URL. A successful login creates a connection to the PPM client server.

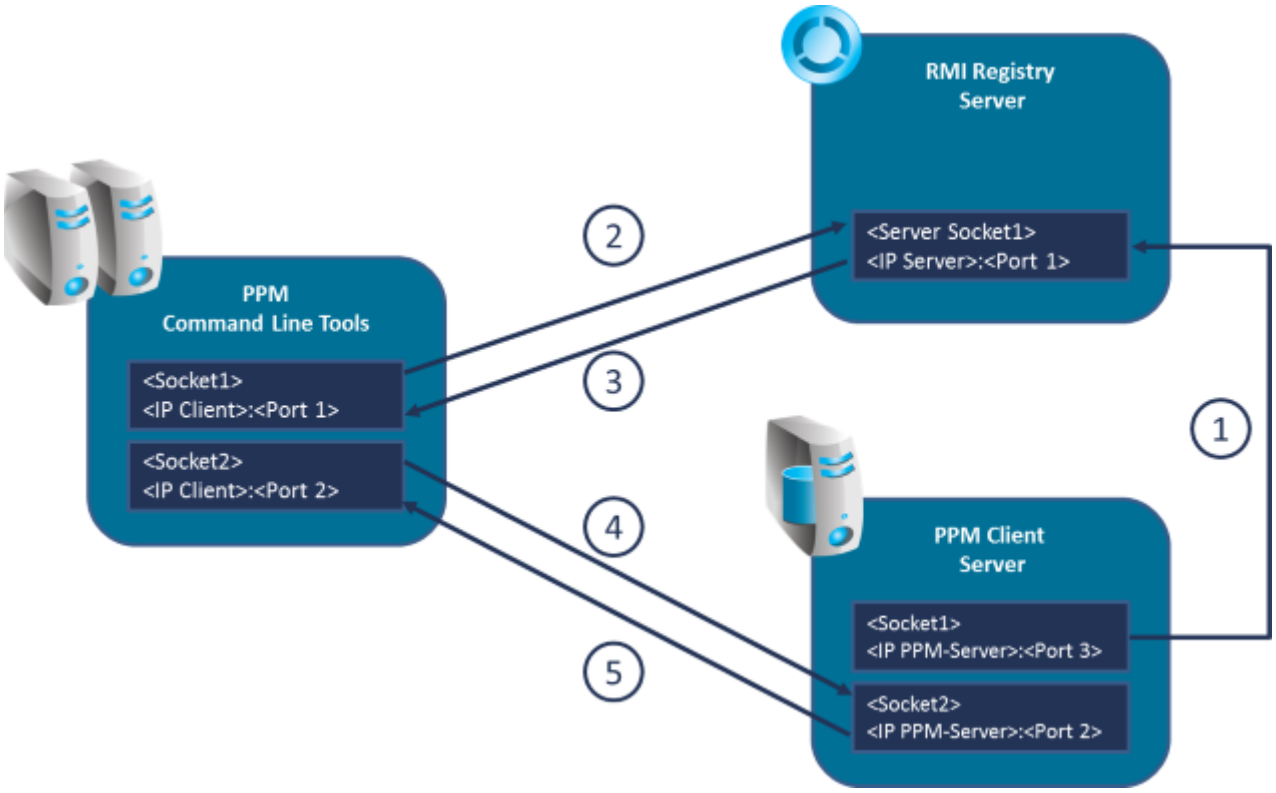
4.4 Data communication between PPM server components

When launched, each PPM client server registers at the RMI registry server with a unique name. This name enables the PPM command line programs and the other server components to query connection information from the registry and to establish a connection to the PPM client server.

The data exchange via RMI is completely transparent for the Java application. All RMI network connections must be available during the entire PPM software runtime.

The RMI objects themselves are designed for direct network communication. The address information contained in the RMI objects (IP address and port number) cannot be evaluated by the firewalls when transferred across network boundaries.

The following figure and table illustrate the RMI data flow described:



	Client	Server	Process
1	PPM client server	RMI registry server	During startup, the client server registers at the registry server with its name.
2	PPM command line programs	RMI registry server	Query whether the relevant client server is available (specification of the client name in the login dialog)
3	PPM command line programs	RMI registry server	The RMI registry returns the RMI object reference of the client server started.
4	PPM command line programs	PPM client server	The front-end executes methods of the client server using the RMI object reference provided.
5	PPM command line programs	PPM client server	The client server returns results.

RMI DATA TRANSFER MODES

You can set different modes for RMI communication between PPM server components, which are described in the following chapters. After you created a client, the **Compressed data transfer** mode described in chapter Compressed data transfer (Page 14) is set.

4.5 Data transfer operation modes

Various data transfer modes are possible in communication between PPM command line programs and the PPM client server:

- Direct
- Compressed
- Encrypted

When you start a client server, the RMI SocketFactory specifies the type of data transfer between the PPM server and command line programs. A line of the form "... The server has been started using SocketFactory <name of the factory>." is displayed in the log after booting the server, which notifies you of the type of RMI data transfer used.

These operation modes are described in the following chapters in more detail.

For data transfer settings please see chapter RMI server (Page 42).

4.5.1 Compressed data transfer (default)

To reduce the network bandwidth required, PPM uses compressed data transfer by default after you created a client. This mode is based on direct communication between the PPM command line programs. This operation mode is preset after creation of a client, but it can be changed at a later time.

CONFIGURATION

To enable the RMI data transfer mode with compression, assign the value **false** to the key **UseSSL** in the global configuration file **Registry_settings.properties** and the value **com.idsscheer.ppm.rmi.compress.ZCompressionSocketFactory** to the key **RMISocketFactory** in the client configuration file **RMIServer_settings.properties**.

4.5.2 Direct RMI data transfer

RMI data transfer is direct, no compression, no encryption. This mode is based on direct communication between the PPM server components.

The additional compression is skipped with this type of data transfer. However, this type of data transfer generates a significantly higher network load.

CONFIGURATION

To enable the RMI default data transfer mode, assign the value **false** to the key **UseSSL** in the global configuration file **Registry_settings.properties** and the value **com.idsscheer.ppm.rmi.ZDefaultSocketFactory** to the key **RMISocketFactory** in the client configuration file **RMIServer_settings.properties**.

4.5.3 Encrypted data transfer

If you want to use SSL technology to encrypt data exchanged between the PPM command line programs and server, please note the following:

- The SSL technology used requires SSL encryption for data communication to always be enabled for the entire PPM system. From a technical point, it is impossible to allow some PPM client servers to access the shared RMI registry in encrypted form, and others to access it in unencrypted form. As a result, SSL encryption is configured across the PPM system.
- When you use the SSL protocol, technical reasons prevent you from also compressing the data exchange.
- Encrypted data communication slightly increases the required computer load.
- Encryption of data communication is recommended only for the PPM server's default operation mode. For performance reasons, we do not recommend using SSL technology in a scaled (distributed) PPM system.

CONFIGURATION

You enable this operation mode by assigning the value **true** to the **UseSSL** key in the global configuration file **Registry_settings.properties**. The configuration of SSL encryption is described in the chapters on Registry (Page 20) and SSL (Page 21). Further information can be found in the **PPM Operation Guide** (chapter **Security aspects/PPM server)**.

If encrypted data transfer is enabled, the **RMISocketFactory** parameter in the client configuration file **RMIServer_settings.properties** is ignored and replaced by an SSLSocketFactory. Usage of this operation mode is to be verified only in the log file of the PPM client server.

Please note that a separate key pair must be generated for SSL encryption for encrypted data transfer. For this, the parameters **PATH_TO_KEYSTORE** and **KEYSTORE_PASSWORD** must be specified in the **SSL_settings.properties** file.

5 Distributed PPM server systems

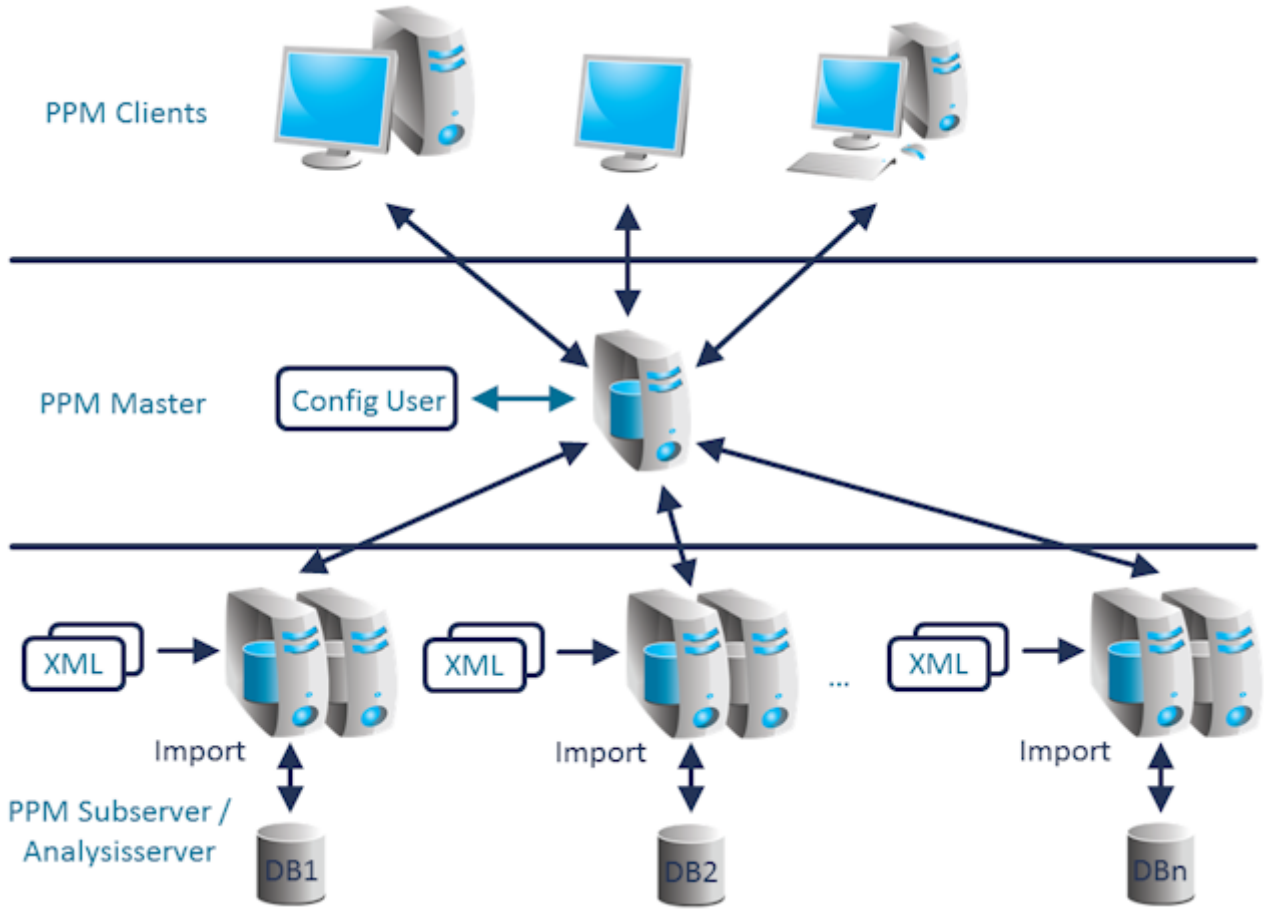
To improve import and analysis performance, you can operate several PPM servers in parallel. These parallel servers are managed by a central master server.

5.1 Access to a distributed system

The PPM front-end communicates exclusively with the master server. The master server passes on queries to the connected sub-servers. The calculation of analysis results takes place simultaneously on the sub-servers. The master server aggregates the results from the sub-servers and sends the overall result to the front-end. To achieve an optimum parallelization result, all sub-servers should, as far as possible, be evenly utilized. Overall system performance is determined by the slowest sub-server.

System configurations are imported using the master server. The master server passes them on to the sub-servers. All PPM servers have the same configuration upon completion of the configuration process.

Example: Data flow between front-end and scaled server system



5.2 Data import in distributed systems

To set up a distributed system, you need to divide the source system data into data packages using a suitable criterion. One data package is associated with one sub-server. Please ensure that

associated process instance data need to be assigned to the same sub-server. If you are using process hierarchies, make sure that the sub-servers each contain the complete tree branches of the hierarchy.

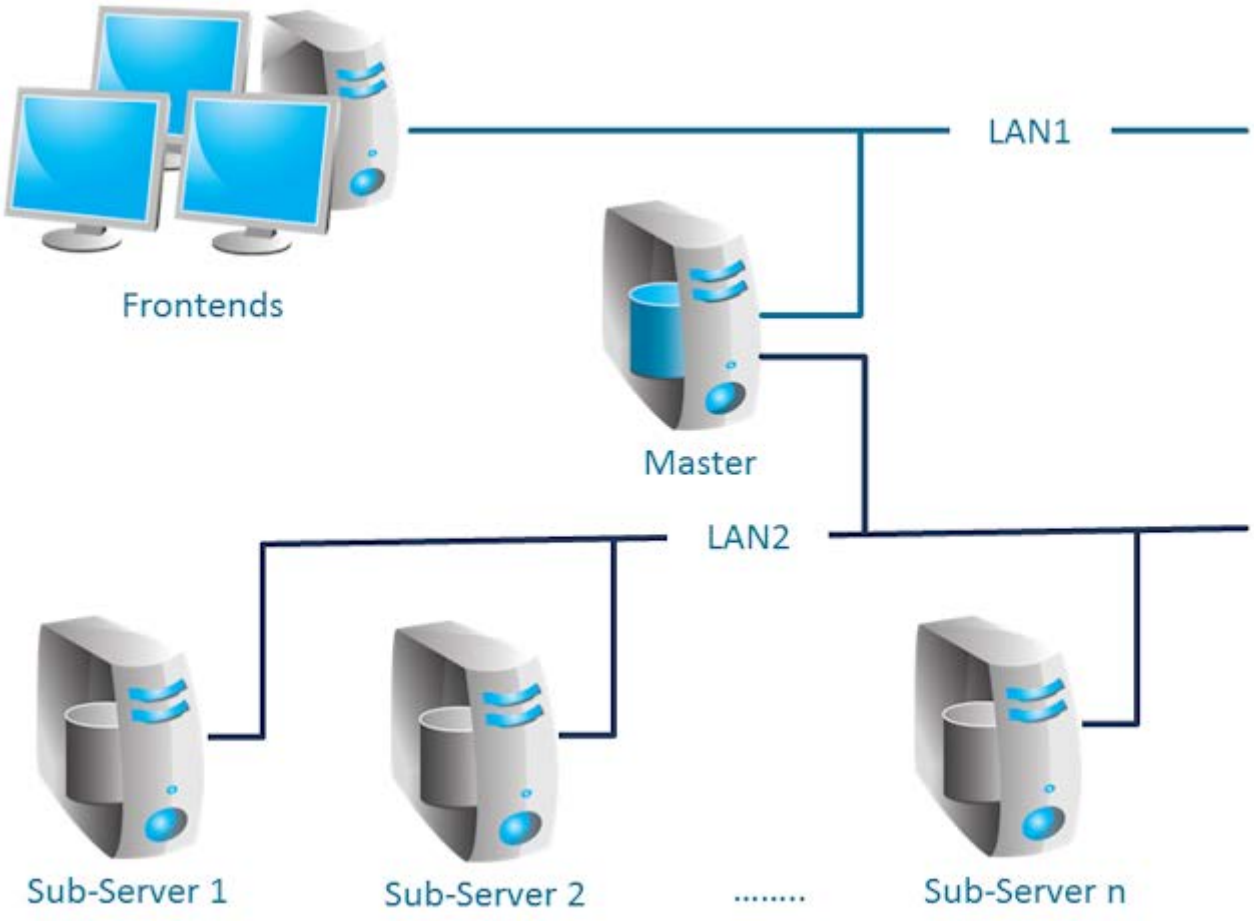
Which criterion you select depends on the use case. For example, the process of dividing the data packages could include dimensions such as the location: Sub-server 1 receives the data from location A, sub-server 2 the data from location B, etc. The **Point in time** criterion is rather unsuitable, though, because it usually limits parallelity of analysis queries and imports.

The data packages are imported directly into the relevant sub-servers. You can select the time of import for each sub-server. Import can, for example, be executed on multiple servers simultaneously in order to reach maximum parallelity.

5.3 Structure of a distributed system

To ensure optimum performance, all sub-servers should use their own system resources (CPU, RAM, hard disk). In addition, the master server should be connected to the sub-servers via a high-speed network. To ensure fault-free communication between the PPM servers in scenarios with a high analysis volume, the network used by the master server to communicate with the sub-servers should be uncoupled from the rest of the network.

Example: Dedicated high-speed network



In principle, installation of multiple sub-servers on a multi-processor system is possible. However, when designing the scaling concept, you need to make sure that the scaling effect is not reduced by shared resources. If a shared database instance is used for multiple sub-servers, each sub-server should be assigned individual data files on physically different data carriers.

5.4 Initializing a scaled system

First, you need to install all sub-servers and then the master server. All sub-servers are specified in the master server's client setup (client setup dialog **Server settings**, section **Operation mode**).

Procedure

1. Initialize all sub-servers first. All sub-servers and the master server use the same configuration file to initialize the PPM languages. The configuration file is specified in the **Initdb_settings.properties** file of each server (see chapter Initdb (Page 36)). The remaining configuration files of the overall system are specified in the **InitSystem_settings.properties** file of the master server (see chapter InitSystem (Page 36)). The sub-servers are configured by the master in step 1. This file is automatically specified by the client setup for all sub-servers and is empty.
2. Next, initialize the master server. The master server imports the system configurations specified in the **InitSystem_settings.properties** file and copies them to the sub-servers.

5.5 Expanding a scaled system

A scaled system can be expanded at any time by adding further sub-servers. The master server can manage a maximum of 64 sub-servers.

Procedure

1. With CTK, a sub-server can be added to an existing system. To do so, change to the **Server settings** dialog in the master server client setup and open the **Change operation mode** dialog in the **Operation mode** section. Use the **Add** button in the **Sub-server specification** dialog to add more sub-servers. Then, set up the new sub-servers as follows.
2. First, use the PPM **runppmconfig** system command to export the entire configuration (including the associated **InitSystem_settings.properties** file) of an existing sub-server.

   Example
   ```
   runppmconfig -user system -client sub1 –mode export -system <installation directory>\ppm\server\bin\work\data_ppm\custom\sub2\xml
   ```
3. Then, copy the file created by runppmconfig (**InitSystem_settings.properties**) into the config directory of the newly created sub-server so that the exported configuration is applied.
4. Now the new sub-server is initialized with the configuration files you exported (PPM system command **runinitdb**). This ensures that the new sub-server has the same configuration as the master server.
5. Restart the master server.

The existing sub-servers do not need to be restarted. The file **InitSystem_settings.properties** copied in step 2 needs to be emptied after initialization of the newly added sub-server in its config directory. The file must be empty.

5.6 Restrictions

5.6.1 Supported scenarios

The scaled system has been specially developed for process-based scenarios. It cannot be used in combination with Data analytics.

The use of process instance-independent measures (PIKIs) is possible in general. However, they cannot be distributed over multiple sub-servers. The sub-server to be used for the PIKIs must be specified during master server configuration.

5.6.2 Query types

Most PPM measure types and comparison values can be used in the master/sub-server system without changes. Statistical calculations (standard deviation, percentiles) and volume-based dimension queries (cardinality measures) are not supported or return valid results only under specific circumstances.

Frequencies cannot be displayed in the interactive filter component.

6 Configure the PPM system

6.1 PPM configuration files

All PPM configuration files are text files that can be edited in any text editor. The configuration file names are composed according to the pattern **<PPM component>_settings.properties**. The home directory for all configuration files is the **config** directory of the PPM installation. The **config** directory contains the global configuration files **Registry_settings.properties** and **SSL_settings.properties**. The subdirectories contain the client-specific configuration files.

6.1.1 Structure

All configuration files have the same structure and the configuration entries are line-based. They have the format **KEY = VALUE**. The **VALUE** value is assigned to the **KEY** configuration characteristic. **KEY** may be divided into different configuration classes by dots (.), for example, <PPMcomponent>.<configuration characteristic>. VALUE may be an actual value in the form of a number or file name, a component-specific keyword or a switch with the value **true** or **false**. Lines that start with the **#** sign are comment lines. Blank lines are permitted.

There are different types of configuration files:

- Global configuration files impact the entire PPM system.
- Client-specific configuration files influence the respective client.

If the global configuration files are modified, it is necessary to restart the RMI and Corba registry server (PPM Core), and all PPM client servers.

Please ensure that you spell keywords in the correct case.

6.1.2 Global configuration files

Data stored in the global configuration files is interactively entered by the user during installation and applies to all clients in the PPM system.

6.1.2.1 Registry

The two entries **RMILeaseValue** and **RMICheckInterval** configure regular checks of the RMI connection in milliseconds and should only be modified in exceptional cases.
The entry **RMIHandshakeTimeout** indicates the time in milliseconds that an RMI client waits for the response of an RMI server before it throws an exception in case of a failed connection and aborts the connection. The setting globally applies to the RMI communication between all PPM components of all clients.

By removing the entry from the file or setting the value to **less than 0** you can restore the standard Java behavior. If you set the value to **0**, no timeout at all occurs. We recommend a value of **15 minutes** (=900000ms) for limiting the time for failed connections.
The **UseSSL** key determines whether an unencrypted protocol – value = **false** – or the **Secure**

Socket Layer (**SSL**) protocol – value = **true** – is used for RMI data transmission in the PPM system. Additional SSL encryption settings are specified in the **SSL_settings.properties** file.

The **CorbaServerActivationPort** key specifies the port that can be used to transfer the runtime information and control instructions for Corba objects.

Both the RMI and Corba registry servers start their services under the IP addressed assigned by the DNS system to the specified computer name.

The ports are set during the installation and are **17500** for the RMI registry and **17590** for the CORBA registry. The values for the host name and the ports can be changed using Cloud Controller. Details can be found in the **PPM Operation Guide** in the **Administration/Configuration** chapter.

PPM client servers that have been started create RMI objects, and if necessary, Corba objects, and register these objects in the registry servers with the port numbers that were specified in the client-specific settings files **RMIServer_settings.properties** and **Corba_settings.properties**.

If multiple network cards are installed in the PPM server computer, specify the IP address of the desired network card, in whose network you want the PPM to be available, instead of the computer name. You can specify the setting using Cloud Controller. Details can be found in the **PPM Operation Guide** in the **Administration/Configuration** chapter.

You can change the URLs for the Corba and RMI registry server using Cloud Controller. Details can be found in the **PPM Operation Guide** in the **Administration/Configuration** chapter.

6.1.2.2 SSL

If you have enabled the encrypted RMI data transfer in the **Registry_settings.properties** file, configure additional SSL encryption settings in the **SSL_settings.properties** file.

The **PATH_TO_KEYSTORE** key specifies the keystore to be used. A keystore is a database file in which both public and private key pairs are saved in an already encrypted form.

The **KEYSTORE_PASSWORD** key specifies the password used to encrypt the keystore.
In the keystore, you only specify a single public-private key pair to be used for the SSL connection. This ensures that a suitable certificate is always available to establish the SSL connection.

6.1.2.3 Kerberos

Kerberos is an authentication protocol that enables mutual authentication of applications in a network by means of key cryptography. PPM is able to apply the user's login data for the operating system (for example, MS Windows) to automatically log the user in to PPM in the Web browser.

KERBEROS PARAMETERS

Use the following entries in central user administration to configure Kerberos access.

- com.aris.umc.kerberos.active

- com.aris.umc.kerberos.config
- com.aris.umc.kerberos.debug
- com.aris.umc.kerberos.kdc
- com.aris.umc.kerberos.keyTab
- com.aris.umc.kerberos.realm
- com.aris.umc.kerberos.servicePrincipalName

By default, PPM uses the MS Windows® native SSPI-API to perform a Kerberos authentication. In case that the usage of the SSPI-API causes incompatibilities you are able to switch the Kerberos authentication to the Java®-internal GSS-API.

To use the Java®-internal GSS-API you can edit the **Kerberos_settings.properties** file and set the **DISABLE_NATIVE_PROVIDERS=** parameter to **true**.

The **Kerberos_settings.properties** file is located in the **<PPM installation>\server\bin\work\data_ppm\config** folder.

6.1.3 Client-specific configuration files

The client-specific configuration is stored in configuration files in a subdirectory whose name corresponds with the client name.

6.1.3.1 AdapterConfig

For the XML import, the **XML_DATEFORMAT** and **XML_DATEFORMAT_ALTERNATIVE** keys specify the time format expected in the log files (specification of date and time). The **XML_TIMEOFDAYFORMAT** and **XML_TIMEOFDAYFORMAT_ALTERNATIVE** keys specify the time (clock time) format expected in the log files. The **XML_DAYFORMAT** and **XML_DAYFORMAT_ALTERNATIVE** keys specify the format of a calendar day expected in the log files.
The format specifications are used with the **TIME** (**DATE)** or **DAY** data type when assigning a source system attribute value to a PPM attribute.

The format settings in the **AdapterConfig_settings.properties** file relate only to the XML import in graph format and the Data analytics import.

6.1.3.2 AnalysisServer

The data in this file is used for analysis server settings. The following table lists the most important setting options:

Key	Example value	Description
RECOVERY_FOLDER	C:\SoftwareAG\ppm\server\bin\work\data_ppm\recovery\umg_en\	Path of the analysis server recovery files (default: <installation directory>\ppm\server\bin\work\data_ppm\recovery\<client>)

Key	Example value	Description
INDEX.PROCESS. <**internaldimname**>. USE	INDEX.PROCESS. MATERIAL. USE=true	Optional configuration of an additional index. Use true to create an index on the dimension with its internal name (<internaldimname>). Only these process dimension types are allowed: Text dimension (one-level, two-level, n-level) Time dimension (timedim) Time of day dimension (hourdim)
INDEX.PROCESS. <**internaldimname**>. REFINEMENT	INDEX.PROCESS. MATERIAL. REFINEMENT= BY_LEVEL2_5	Additional, optional information on the created index of a dimension (only with one-level text dimensions). Refinement that the index is to be specified with. Default value is the roughest value (BY_LEVEL_1, BY_YEAR, BY_HOUR_OF_DAY). Valid values: Two/level text dimension BY_LEVEL1, BY_LEVEL2 N-level text dimension BY_LEVELX_Y, with X indicating the selected refinement level and Y indicating the maximum number of levels of the dimension (with the example value BY_LEVEL2_5, an index is created on the second level of the five-level text dimension MATERIAL) Time dimension, valid values: BY_DAY, BY_MONTH, BY_QUARTER, BY_YEAR Time of day dimension, allowed values: BY_MINUTE_OF_DAY, BY_HOUR_OF_DAY Only one index is allowed for each dimension. If multiple indices are activated or deactivated or refinement data is configured for one and the same dimension, only the last data is used. Wrong dimension types are ignored without error message output. Once an index has successfully been created, the corresponding information is output.

Key	Example value	Description
RETRIEVER_MAX_ INSTANCES	50000 (default value)	Maximum number of process instances or function/relation instances that the analysis server can query. If the specified value is exceeded in a process instance query, the query aborts with an error message. Larger values result in increased main memory requirement of servers and user interfaces.
RETRIEVER_MAX_ITERATION_STEPS	5000 (Default value)	Maximum number of data rows that can be queried by the analysis server. If the specified value is exceeded in the result, the query aborts with an error message. Larger values result in increased main memory requirement of servers and user interfaces.
RETRIEVER_MAX_FILTER_ITEMS_FOR_XA_TO_PI	10000 (Default value)	Maximum number of values in the filter when including a query from a Data analytics analysis realm in the process analysis. If more values are generated, the query is canceled. Larger values result in increased main memory requirement of servers and user interfaces.
MemoryLoadGuard.Enabled	true	Activation of the memory load guard system
MemoryLoadGuard.Warn.PercentageOfMemoryUsed	90	Memory load based on which an alert is generated.
MemoryLoadGuard.PreventImport.PercentageOfMemoryUsed	95	Memory load based on which no new data imports into the analysis server are allowed.

Key	Example value	Description
MemoryLoadGuard.BackgroundCheck.TimeWindow.Start	22:00	Start time based on which the background check takes place.
MemoryLoadGuard.BackgroundCheck.TimeWindow.End	05:00	End time based on which the background check ends.
LANGUAGE	de (ISO code based on ISO 639-1)	Language that the analysis server is to operated with (according to ISO 639-1; if blank, the operating system default language is used; if that language or the language specified is not supported by the client, the client's default language is used).

6.1.3.3 AnalysisServer_Log

The data in this file is used for analysis server system messages settings.

Detailed information is available in the **PPM Operation Guide**.

6.1.3.4 Chart

The information in this file determines the appearance of the charts generated by the PPM server. You can, for example, define labeling colors and character sets. The file contains comprehensive comments describing the effects of the entries.

6.1.3.5 CNet (Communication Net)

The information in the file reflects the default values of the individual keys, which influences the appearance of a communication network created in the **Interaction analysis** module. All parameters are optional. The file contains comprehensive comments describing the effects of the entries.

Key	Description
DEFAULT_NODE_ MODE	Node representation mode. Valid values: **COMPLETE_MODE** (organizational unit in ARIS notation) or **SIMPLE_NODE** (node as color shaded circle). Default value: **COMPLETE_MODE**
DEFAULT_NODE _SIZE	Default diameter of circle (only in **SIMPLE_NODE** mode)
DEFAULT_EDGE_ THICKNESS_MODE	Connection weight representation mode depending on: - Percentage share of outgoing connections (**OUTGOING_MODE**) - Percentage share of incoming connections (**INCOMING_MODE**) - Absolute measure value (**VALUE_MODE**) Representation of uniform connection weights: **NONE_MODE**
DEFAULT_ EDGE_VALUE_MODE	Connection value representation mode depending on: - Percentage share of outgoing connections (**OUTGOING_MODE**) - Percentage share of incoming connections (**INCOMING_MODE**) - Absolute measure value (**VALUE_MODE**) Connection values not represented: **NONE_MODE**
DEFAULT_ EDGE_THICKNESS	Connection weight for uniform representation (only in **NONE_MODE** mode)
DEFAULT_ EDGE_COLOR	Connection color
DEFAULT_ NODE_COLOR	Node color
DEFAULT_ BACKGROUND_COLOR	Background color
DEFAULT_ SELFEDGE_MODE	Visualization of internal communication of a node (measure value) using the diameter of the circle (only in **SIMPLE_NODE** mode). Valid values: - **true** (Show) - **false** (Do not show)
DEFAULT_EDGE_ PIXEL_RANGE	Value range (floating point number) for connection weight in pixels (min, max) for dependent representation (all modes except **NONE_MODE**)
DEFAULT_NODE_ PIXEL_RANGE	Value range for diameter of circle in pixels (min, max) for representation depending on the measure value (**DEFAULT_SELFEDGE_MODE=true**)

Key	Description
UNDER_WARNING_ EDGE_COLOR	Connection color in RGB format for measure values that are better than the warning value specified (= planned value 1 for measure).
WARNING_EDGE_ COLOR	Connection color in RGB format for measure values between the warning value (= planned value 1) and the alarm value (= planned value 2).
ALARM_EDGE_ COLOR	Connection color in RGB format for measure values that are poorer than the alarm value (= planned value 2).
ONLY_FOR_ RELATIONS	Type of analysis of communication network. Valid values: - **true** (default value: only relationships between organizational dimensions) - **false** (between any two dimensions)

6.1.3.6 Corba server

The information in this file configures the data exchange via the Corba protocol. The PPM query interface uses the Corba protocol to communicate with the PPM client server.

The **CorbaObjectsPort** key specifies the port number used by the PPM client server. Corba objects created by the client server are registered with this port number at the Corba registry.

6.1.3.7 Database

The defaults in this file configure database-dependent basic mechanisms of the PPM system and describe the interface between PPM server and database server.

GENERAL PART

Key	Description
DATABASE_TYPE	Specifies the type of the RDBMS to be used. Valid values are **ORACLE 11**, **ORACLE_12**, **DB2_10**, **SQLSERVER_2014**, and **SQLSERVER_2014_UNICODE**, **SQLSERVER_2016**, and **SQLSERVER_2016_UNICODE**.
URL	URL of the PPM databases in JDBC notation. Depends on the database system and DATABASE_TYPE used.
USER	Database user of the PPM databases

Key	Description
PASSWD	Password for the PPM databases. If the password is specified or changed in the CTK client settings it is automatically encrypted. Manual setting of an unencrypted password directly in the database settings, for example for testing purposes, is also supported.
TRUNCATE_DIMENSION_STRINGS/ TRUNCATE_DIMENSION_MARK	Behavior with text-based dimension values that exceed the configured maximum length of the dimension. Valid values are **TRUE**, **FALSE** (default), or **MARK**. FALSE: Values and descriptions are truncated to the configured maximum length of the dimension. The measure calculator outputs an error message. TRUE: as FALSE. However, there will be no error message. MARK: Values and descriptions of dimensions exceeding the maximum length are replaced by the text in TRUNCATE_DIMENSION_MARK. The measure calculator outputs a warning.

For some special cases that might occur during data import, you need to specify default valuesPPM when initializing the system.

Warning

Please note that these values must not be changed in a system that has already been initialized.

Key	Meaning
PPM_NULL	Value used for dimensions that have not been specified or maintained. (UTF-encoded string. The value must be changed before the database is initialized).
ERROR_NODE.STR	Name of the process type group in the process type which contains all incorrectly calculated/typified instances (UTF-encoded string. The value must be changed before the database is initialized).
NOT_TYPIFIED.STR	Name of the process type in the process type, which contains instances that cannot be typified. (UTF-encoded string. The value must be changed before the database is initialized.)
WRONG_TYPIFIED.STR	Name of the process type in the process type, which contains instances that were incorrectly typified. (UTF-encoded string. The value must be changed before the database is initialized.)

TABLESPACES/DATA FILES USED

For performance reasons, it is recommended that you save different data types separately in the database system. The definition depends on the RDBMS used and the installation of the database system. Please refer to the **PPM Database systems** manual for a more detailed description.

Key	Meaning
<DATABASE_TYPE>_TBLCONF_STDTABLE/ <DATABASE_TYPE>_TBLCONF_STDINDEX/ <DATABASE_TYPE>_TBLCONF_STDBLOB	These parameters determine in which tablespaces or data files certain types of data are saved.

INTERNAL PART

The other parameters in the database settings configure the basic mechanisms of the database and PPM. They allow specific adjustments and fine-tuning in special constellations (for example, in case of different RDBMS behavior within a main version or for special import scenarios). The default settings usually cover all usage scenarios optimally and should therefore be changed only if absolutely necessary.

Warning

Changes to internal settings can affect the entire database negatively and might permanently damage correctness, performance, and consistency of the PPM system. Please change them only if you are explicitly requested to do so by PPM product support.

6.1.3.8 EPC

SIZE LIMIT OF THE PROCESS INSTANCE TABLE

```
#Limit for instances in instancelist (0 = No question at all)
CRITICAL_INSTANCE_COUNT_TO_VIEW = 15000
```

Maximum number of process instances displayed in a process instance table. If this number is exceeded during a query, the system asks if you really want to run the query. Default value: 15000.

6.1.3.8.1 Adjustment of the EPC representation

DEFAULT SETTING FOR EPC VIEW

```
#Configuration of the default view
#Valid values: epk, attributes, functions, gantt
Default_Epk_View = epk
```

View to be shown when opening an EPC. If no value is specified or the entry is missing, the **epk** view is selected as a default.

COLOR INTENSITY OF FUNCTIONS

```
#Configuration of color intensities:
FUNCTION_INTENSITY_MAX.0 = 0.2
```

```
FUNCTION_INTENSITY_MAX.1 = 0.4
FUNCTION_INTENSITY_MAX.2 = 0.6
FUNCTION_INTENSITY_MAX.3 = 0.8
```

For the display of aggregated EPCs, these settings configure the color intensity depending on the number of executions of the respective functions. For each of the four levels, the threshold value is specified as a factor.

CONNECTION CONFIGURATION

```
#Configuration of connection categories:
EDGE.0 = EDGE1
EDGE.1 = EDGE2
EDGE.2 = EDGE3
EDGE.DEFAULT_WITH = 2

EDGE1.max = 0.3
EDGE1.width = 1

EDGE2.max = 0.7
EDGE2.width = 3

EDGE3.max = 1
EDGE3.width = 5
```

When displaying aggregated EPCs, these settings configure the weight of flow connections depending on the number of the connection's executions. For each of the three levels, the threshold is specified as a factor and the connection weight in pixels.

LAYOUT CONFIGURATION

Warning

The following settings originate from the layout algorithm for ARIS Business Architect and should not be changed.

Setting	Value	Description
LAYOUT_ALGO_NO	20	Layout algorithm number
LAYOUT_EPK_ALGO	1	Layout procedure for the non-aggregated EPC view
LAYOUT_EPK_ ALGO_COMPRESSED	1	Layout procedure for the aggregated EPC view
LAYOUT_LONGEST_ PATH_POSITION	0	Alignment of the longest path: centered, right, left
LAYOUT_BREAK_SPACE	true	Replace space with line break
LAYOUT_MAKE_SPACE	true	Create space in partial layout
LAYOUT_ROOT_DOWN	1	Root position relative to the subtree
LAYOUT_ROOT_POSITION	0	Root node position
LAYOUT_CHANGE_ SON_ARRANGEMENT	2	Change to vertical layout
LAYOUT_ ARRANGE_SATTELITES	false	Arrange satellites

Setting	Value	Description
LAYOUT_OBJECT_SCALING	false	Scale objects
LAYOUT_SHOW_TRIMMED	false	Display untrimmed EPC
LAYOUT_SHOW_ ATTRIB_TITLE	false	Display attribute title
LAYOUT_XSPACING	30	Horizontal object spacing
LAYOUT_YSPACING	50	Vertical object spacing
LAYOUT_FONTSIZE	10	Font size in EPC objects
LAYOUT_HORIZONTAL	false	EPC is not represented in horizontal layout

PLACING OBJECT ATTRIBUTES

```
#Configuration of default attribute placements:
EPK_NODE_ATTRIBUTE_PLACE_CENTER      = AT_OBJNAME
EPK_NODE_ATTRIBUTE_PLACE_RIGHT_TOP   = AT_AV_PROC_TIME
EPK_NODE_ATTRIBUTE_PLACE_RIGHT_BOTTOM =
EPK_NODE_ATTRIBUTE_PLACE_LEFT_BOTTOM  = AT_END_TIME
EPK_NODE_ATTRIBUTE_PLACE_LEFT_TOP    = AT_START_TIME
```

Includes information about the position of individual object attributes in the EPC view. The data is specified in the following form:

EPK_NODE_ATTRIBUTE_PLACE_**<Position> = <Identifier of the attribute>**

DISPLAY OBJECT ATTRIBUTES IN TOOLTIPS

```
#Configuration of default tooltip display
SHOW_TOOLTIP.0                  = AT_PROCTYPEGROUP
SHOW_TOOLTIP.1                  = AT_PROCTYPE
```

Determines the attribute values that are displayed when the mouse pointer is positioned over an object. To display more attributes, you can add further key value pairs to the list. The numeric part of the key filed must be increased accordingly.

Only consistently numbered list entries will be considered.

6.1.3.8.1.1 Function-specific adjustments

In addition to configuring the display of general object attributes it is possible to adjust them in terms of function.

The relevant functions must be known to the system. This is implemented with the keyword:

`FUNCTION.0  = <function name>`

Further key value pairs can specify additional functions. The numeric part of the key filed must be increased accordingly.

Only consistently numbered list entries will be considered.

You can then specify tooltips and placed object attributes for the functions thus defined.

The following example describes this for the **SAP.AUFT** function:

```
FUNCTION.0                      = SAP.AUFT
```

DISPLAY OBJECT ATTRIBUTES IN TOOLTIPS (FUNCTION-SPECIFIC)

```
#Configuration of default tooltip display
SAP.AUFT.SHOW_TOOLTIP.0                  = AT_PROCTYPEGROUP
SAP.AUFT.SHOW_TOOLTIP.1                  = AT_PROCTYPE
SAP.AUFT.SHOW_TOOLTIP.2                  = AT_KI_FEDFREQ
```

Determines the attribute values that are displayed when the mouse pointer is positioned over the relevant function. To display more attributes, you can add further key value pairs to the list. The numeric part of the key filed must be increased accordingly.

Only consistently numbered list entries will be considered.

PLACING OBJECT ATTRIBUTES (FUNCTION-SPECIFIC)

```
#Configuration of attribute placements for function 'Create customer order'
SAP.AUFT.EPK_NODE_ATTRIBUTE_PLACE_CENTER      = AT_OBJNAME
SAP.AUFT.EPK_NODE_ATTRIBUTE_PLACE_RIGHT_TOP    = AT_AV_PROC_TIME
SAP.AUFT.EPK_NODE_ATTRIBUTE_PLACE_RIGHT_BOTTOM =
SAP.AUFT.EPK_NODE_ATTRIBUTE_PLACE_LEFT_BOTTOM  = AT_END_TIME
SAP.AUFT.EPK_NODE_ATTRIBUTE_PLACE_LEFT_TOP     = AT_START_TIME
```

Determines the attribute values to be displayed at an EPC node. The key consists of the key of the function (in this example: SAP.AUFT) and the position **EPK_NODE_ATTRIBUTE_PLACE_<Position>**.

6.1.3.8.2 EPC aggregator settings

TYPE OF FUNCTIONS AGGREGATION

```
#Should orgunit attributes at functions be used for compression?
EPK_USE_ORGUNIT_FOR_COMPRESS = false
```

If the value of this setting is **true**, organizational units are taken into account when aggregating process instances. Identical functions that are assigned to different organizational units are handled as different functions. Default value: **false**

CALCULATION OF THE PERSISTENT AGGREGATOR

```
# Number of instances marked for deletion before a KI calculation is initiated
COMPRESSOR_KICALCULATION_THRESHOLD=100000
```

The persistent aggregator works in packages comprising two phases each. During the first phase, the process instances to be aggregated are merged and saved as a new EPC. Afterwards, the measure calculator calculates the new instances and consolidates the data. This parameter determines the size of the packages, that is, the number of source instances after which the system changes to the **measure calculation** phase. Default value: **100000**

DELETING HIERARCHICAL PROCESS INSTANCES

```
#Default behavior when deleting hierarchical EPCs
DELETE_REFERENCED_EPCS_WHEN_DELETETED = false
```

If the value of this setting is **true**, the assigned process instances of all hierarchy levels are deleted when deleting process instances. Default value: **false**

```
#Default behavior when compressing hierarchical EPCs
DELETE_REFERENCED_EPCS_WHEN_COMPRESSED = false
```

If the value of this setting is **true**, the assigned process instances of all hierarchy levels are deleted when persistently aggregating process instances. Default value: **false**

Warning

When aggregating or deleting process instances, assigned process instances are deleted regardless of process access privileges. PPM users with the **Process instance aggregation** function privilege can also delete process instances for which they usually do not have access privileges.

AGGREGATION OF TIME RANGE DIMENSIONS

```
#Delete rangedimension values for compression
DELETE_RANGEDIMENSION_VALUES_FOR_COMPRESSION = false
```

This value determines the behavior during persistant aggregation of processes in combination with time range dimensions. Persistent aggregation of process instances keeping the time range dimensions is impossible. If time range dimensions are defined for relevant process instances, the attempt to persistently aggregate is aborted by default (default value: **false**) with an error message. If the value of this parameter is set to **true** , an aggregation takes place even if time range dimensions exist. Time range dimensions are not included in aggregation so that their values are no longer available at an aggregated instance.

6.1.3.9 EPC import

These parameters control mechanisms used by PPM during data import.

The import scenarios **Small**, **Medium**, and **Large** are provided by CTK from PPM 10.2 onwards. Depending on the selected scenario in the **IMPORT_SCENARIO** parameter (for example, IMPORT_SCENARIO=SMALL) the appropriate parameters are automatically taken (for example, SMALL_READE_RATE_EPC=100000). The **Small** import scenario contains the default parameter values. See the documentation **PPM Data Import** for details.

Key	Description
<IMPORT_SCENARIO>_READ_RATE_EPC	To limit the main memory requirement necessary for structural information, PPM import packages the fragment files. These packages are processed in individual runs of the import component. The value stated corresponds to the package size. By increasing the value, the overhead that is generated by the individual processing of the components (for example, multiple calculation of individual instances) can be reduced. However, it is important that enough main memory is available for PPM import to store the larger volume of structural information. If required, the JVM parameters of the PPM import need to be adjusted for these clients in CTK. Default value (SMALL_READ_RATE_EPC): **100000**

Key	Description
<IMPORT_SCENARIO>_XML_IMPORT_COMMIT_RATE	Number of fragments that are written to the database in one transaction by the XML import. Default value (SMALL_XML_IMPORT_COMMIT_RATE): **50000**
<IMPORT_SCENARIO>_EPC_IMPORT_COMMIT_RATE	Number of fragments processed in the "EPC import" phase during PPM import in one transaction from the database. Default value (SMALL_EPC_IMPORT_COMMIT_RATE): **25000**
<IMPORT_SCENARIO>_XML_IMPORT_WRITE_BUFFER	During XML import, the fragments are saved permanently in the database by a simultaneously running background process. To balance speed fluctuations of the import or RDBMS, a buffer is used. The parameter controls the maximum number of fragments that are cached in this buffer. If the bugger size is reduced to **0**, the background process is deactivated. Default value (SMALL_XML_IMPORT_WRITE_BUFFER): **75**
<IMPORT_SCENARIO>_KI_EPC_FUNCTION_COUNT_THRESHOLD	This parameter is used when handling large EPCs. It specifies the maximum number of functions based on which a process instance is considered to be "of normal size". Details about this mechanism are described in the **PPM Data import** manual. Default value (SMALL_KI_EPC_FUNCTION_COUNT_THRESHOLD): **500**

PPM PREMERGE MECHANISM

Premerges (XML and EPC) combine individually imported fragments in a runtime-based cache before saving them permanently in the database. This significantly reduces the number of database objects and increases the efficiency of the import and subsequent phases.

The effectiveness of the premerge is considerably influenced by the order of system events within the input files. If possible, associated events, such as events with identical process keys, should follow each other as closely as possible in the input files.

PPM uses the XML premerge as a default. It is used during the XML import. It processes the fragments at the earliest point in time possible.

The EPC premerge is deactivated by default. It can be used if during XML import many small individual imports run so that a combination of fragments by the XML premerge is impossible.

An increase of the premerge cache can be used to increase the area in which associated fragments are identified. Please note that this leads to increased memory requirements of the XML or PPM import. If required, the JVM parameters of the components need to be adjusted for these clients in CTK.

Since the runtime cache is emptied when a database transaction is terminated, the transaction fragment **<IMPORT_SCENARIO>_<XML/EPC>_IMPORT_COMMIT_RATE** (see above) should be adjusted if the premerge cache is significantly increased.

Key	Description
<IMPORT_SCENARIO>_XML_IMPORT_PREMERGER_CACHE_SIZE	Size of XML premerge cache. Default value (SMALL_XML_IMPORT_PREMERGER_CACHE_SIZE): **5000**
<IMPORT_SCENARIO>_EPC_IMPORT_PREMERGER_CACHE_SIZE	Size of EPC premerge cache. Default value (SMALL_EPC_IMPORT_PREMERGER_CACHE_SIZE): **0** (deactivated)

If you specify the value **0** for the cache size, the corresponding cache is disabled.

The premerge settings apply to the XML import (runxmlimport) or PPM import (runppmimport).

6.1.3.10 Help

In this file, you can configure optional, user-defined menu entries for the client, which show more web sites. You can configure individual URLs for different languages. The URLs can be absolute or relative to the client URL (http(s)://host:port/ppm/html/). When using PPM in the cloud, when specifying absolute URLs the host addresses accessible from the Internet must be set. Fallback entries are configured for languages that are not defined.

In the URL, the placeholder **{0}** can be inserted for the locale, it will be resolved when the page is called.

For example, for the URL **http://host:port/help/{0}/help.html** to be displayed in English, the placeholder **{0}** is replaced by **en** and for German with **de**. If the same URL or URL with placeholder is used for all entries, it is not required to configure a URL for each entry.

If all language-related URLs have the same language-independent menu entry, only the fallback entry must be configured.

Key	Meaning
HELP.x.URL	URL of the fallback entry. "x" indicates an index starting with "0" for each menu entry.
HELP.x.MNI	Menu name of the fallback entry.
HELP.x.<language>.URL	URL of a menu entry for any language (the two-letter ISO codes, such as "de" for German apply to "<language>")
HELP.x.<language>.MNI	Menu name of the entry for any language.

Examples

Usage of a placeholder URL and multiple, language-specific entries:

HELP.0.URL = http://host:port/html/help/{0}/custom/KI_HELP.htm

HELP.0.MNI = Measure help

HELP.0.de.MNI = Measure help

HELP.0.fr.MNI = L'aide de l'indicateur de performance

Usage of an entry and individual URLs for different languages:

HELP.1.URL = http://www.softwareag.com

HELP.1.MNI = Software AG

HELP.1.de.URL = http://www.softwareag.de

HELP.1.fr.URL = http://www.softwareag.fr

Relative URLs:

HELP.2.URL = /docs/customizing/index.html

HELP.2.MNI = Customizing Overview

6.1.3.11 Initdb

Controls the first phase of database initialization. The language keys specified in the referenced XML file determine the default language and the possible alternative languages of PPM.

6.1.3.12 InitSystem

This controls the second phase of database initialization in which the customizing files are imported. The file is divided into sections. The specifications of a section are used as arguments for internally performed calls of the **runppmconfig** configuration program. **XXX** corresponds to consecutive numbering.

```
INIT_MODULE_XXX =
INIT_MODULE_XXX_NAME = <name>
```

Name of the configuration component. Corresponds to the **-command** argument of the **runppmconfig** tool.

```
INIT_MODULE_XXX_FILE = <file name>
```

XML file to be used.

6.1.3.13 Kerberos

Kerberos is an authentication protocol that enables mutual authentication of applications in a network by means of key cryptography. PPM is able to apply the user's login data for the operating system (for example, MS Windows) to automatically log the user in to PPM in the Web browser.

KERBEROS PARAMETERS

Use the following entries in central user administration to configure Kerberos access.

- com.aris.umc.kerberos.active
- com.aris.umc.kerberos.config

- com.aris.umc.kerberos.debug
- com.aris.umc.kerberos.kdc
- com.aris.umc.kerberos.keyTab
- com.aris.umc.kerberos.realm
- com.aris.umc.kerberos.servicePrincipalName

By default, PPM uses the MS Windows® native SSPI-API to perform a Kerberos authentication. In case that the usage of the SSPI-API causes incompatibilities you are able to switch the Kerberos authentication to the Java®-internal GSS-API.

To use the Java®-internal GSS-API you can edit the **Kerberos_settings.properties** file and set the **DISABLE_NATIVE_PROVIDERS=** parameter to **true**.

The **Kerberos_settings.properties** file is located in the **<PPM installation>\server\bin\work\data_ppm\config** folder.

6.1.3.14 Keyindicator

Controls the Measure calculator. The table below lists the descriptions of a selection of the most important keys of the file.

Key	Meaning
MAX_STEP_COUNT	Maximum number of iteration steps in the chart display.
MAX_TIME_ STEP_COUNT	Maximum number of iteration steps in the **Time** dimension.
MAX_DATASET_ EXTENDER_SIZE	If no value can be determined for measures of the **NUM_KEYINDICATOR** and **FREQ_KEYINDICATOR** retriever type, the result set of the measure query is stated as 0. Result sets up to the size specified here are taken into account.
USE_KI_CACHE	**true** activates the buffer of the precalculated favorites.
AUTO_FILL_CACHE	**true** triggers automatic filling of the cache after deletion (only for USE_KI_CACHE=true).
FILL_CACHE_ DELAY_TIME	Delay in milliseconds before precalculation of favorites starts after finishing a PPM data import.
KI_LRU_ CACHE_SIZE	Number of queries saved in the main memory based LRU (least recently used) measure cache. Recommended value: **50**. The value **0** disables this option.
EPK_LRU_ CACHE_SIZE	Number of EPC queries saved in the main memory based LRU EPC cache. Recommended value: **50**. The value **0** disables this option.

Key	Meaning
DEFAULT_ LIMITPERC	Predefined threshold value (in percent) for displaying analysis results in the Process Mining Wizard, above which unfavorable deviations from the measure value or conspicuous variation limits of dimension values are to be displayed. Default value: **10.0%**
DEFAULT_ RELEVANCEPERC	Specified threshold value (in percent) in terms of the relevance of dimension values for the display of analysis results in the Process Mining Wizard, above which dimension values are to be displayed. Default value: **10.0%**
DEFAULT_ NUMBEROFLINES	Maximum number of result lines per process type in the Process Mining Wizard. Default value: **15**
DEFAULT_DIMENSION_LEVEL_DELIMITER_IN_URL	Separator between dimension levels
OUTLIER_SIGMA_VALUE	Sigma value (factor) for calculating the limit value in a default calculation
ORGUNITS_MAX_STRING_LENGTH	Maximum length of the name of an organizational unit (default value: **100**); is used in the Administration component of organizational units.
FAVORITE_CACHE_LOGFILE	Name of the log file for favorites calculation; empty indicates that the log is deactivated (default).
FAVORITE_CACHE_LOGFILE_THRESHOLD	Minimum execution time (in seconds) that a log causes (default: **0**, all calculations are logged). If FAVORITE_CACHE_LOGFILE is not specified, this value is ignored.
FAVORITE_CACHE_LOGFILE_PARAMSET_THRESHOLD	Minimum execution time (in seconds) that causes additional logging of the underlying paramset (default: **-1**, no additional logging). If FAVORITE_CACHE_LOGFILE is not specified, this value is ignored.
FAVORITE_CACHE_LOGFILE_LOG_CALCULATION_START	Value **TRUE** or **FALSE**. With TRUE, a log entry is written every time a favorites thread starts a new calculation.

6.1.3.15 Mail

The computer specified in the **EMAIL_SERVER** key (TCP/IP network name or IP address) receives the e-mails sent by the PPM server via the SMTP protocol.

The e-mail address specified in the **EMAIL_FROM** key is used as sender address for the following types of e-mail:

1. Sender address for e-mails that are generated by automation

2. Sender address for e-mails that are generated by report automation, if no e-mail address was specified for the executing user or if an e-mail address cannot be determined for any other reason
3. Sender address for e-mails that are generated from the **Actions** module, if no e-mail address was specified for the executing user or if an e-mail address cannot be determined for any other reason
4. Sender address for e-mails that are generated by the **runppmanalytics** program, if no e-mail address was specified for the executing user or if an e-mail address cannot be determined for any other reason

The **FORMAT** and **STYLE** keys specify the formats to be used in the Activities table and Messages table. These formats must be defined in the **Mail_settings.properties** file.

```
REPORT_DETAIL_ACTIVITY_STYLE = null
REPORT_DETAIL_ACTIVITY_FORMAT = cpi_detail_html

REPORT_DETAIL_COMMENT_STYLE = null
REPORT_DETAIL_COMMENT_FORMAT = cpi_detail_html

REPORT_EMAIL_ACTIVITY_STYLE = null
REPORT_EMAIL_ACTIVITY_FORMAT = cpi_detail_plaintext

REPORT_EMAIL_COMMENT_STYLE = null
REPORT_EMAIL_COMMENT_FORMAT = cpi_detail_plaintext
```

The keys **REPORT_EMAIL_ACTIVITY_TYPE** and **REPORT_EMAIL_COMMENT_TYPE** indicate the format of the e-mail created. Valid values are **text/plain** for e-mails in text format and **text/html** for e-mails in html format.

```
REPORT_EMAIL_ACTIVITY_TYPE = text/html
REPORT_EMAIL_COMMENT_TYPE  = text/html
```

Use the key **EMAIL_ATTACH_RESULT** to specify whether and how an analysis linked to a message will be attached. Valid values:

Key value	Description
false	The linked analysis will not be attached to the message.
inline	The linked analysis will be embedded in the message. The message type is automatically set to text/html.
pdf	The linked analysis will be attached to the message as a pdf file.

The analysis linked and attached to a message is created using the styles specified in the keys **EMAIL_INLINE_STYLE** or **EMAIL_PDF_STYLE**.

Use the **SEND_ANALYTICS_EMAILS** key to indicate whether an e-mail will be sent by default (value = **true**) or not (value = **false**) in addition to creating a CPI message if unfavorable deviations exist.

```
SEND_ANALYTICS_EMAILS = false
```

For each Easy mining message type (Early alert, planned value, alarm value, suspected deviation, and outlier analyses), you can use key values to specify whether you are notified by e-mail (key

value **EMAIL**), by a message in the **Improvements** module (key value **CPI**), or both (key value **CPI,EMAIL**).

```
TARGETVALUE_BEHAVIOUR=CPI
PROCESSMINING_BEHAVIOUR=CPI
ALARMVALUE_BEHAVIOUR=CPI
EARLYALERT_BEHAVIOUR=CPI
OUTLIER_BEHAVIOUR=CPI
```

If you do not specify any key value, a message is created in the **Improvements** module and, depending on the **SEND_ANALYTICS_EMAILS** key value, an e-mail may also be sent (value = **true**) or not (value = **false**). If you specify a key value, the information in the **SEND_ANALYTICS_EMAILS** key is ignored.

Warning

The values for **EMAIL** and **CPI** are case-sensitive. Incorrect data results in the action not being executed.

6.1.3.16 RE (Relation Explorer)

The information in the file reflects the default values of the individual keys that influence the appearance of a Relation Explorer chart. All parameters are optional. The file contains comprehensive comments describing the effects of the entries.

Key	Description
DEFAULT_NODE_MODE	Node representation mode. Valid values: **COMPLETE_MODE** (organizational unit in ARIS notation) or **SIMPLE_NODE** (node as color shaded circle). Default value: **COMPLETE_MODE**
DEFAULT_NODE_SIZE	Default diameter of circle (only in **SIMPLE_NODE** mode) for measure-independent representation (**DEFAULT_SELFEDGE_MODE=false**)
DEFAULT_EDGE_THICKNESS _MODE	Connection weight representation mode depending on: - Percentage share of outgoing connections (**OUTGOING_MODE**) - Percentage share of incoming connections (**INCOMING _MODE**) - Absolute measure value (**VALUE_MODE**) Representation of uniform connection weights: **NONE_MODE**
DEFAULT_EDGE_VALUE_MODE	Connection value representation mode depending on: - Percentage share of outgoing connections (**OUTGOING_MODE**) - Percentage share of incoming connections (**INCOMING _MODE**) - Absolute measure value (**VALUE_MODE**) Connection values not represented: **NONE_MODE**
DEFAULT_EDGE_THICKNESS	Connection weight for uniform representation (only in **NONE_MODE** mode)
DEFAULT_EDGE_COLOR	Connection color in RGB format

Key	Description
DEFAULT_NODE_COLOR	Node color in RGB format
DEFAULT_BACKGROUND_COLOR	Background color in RGB format
DEFAULT_SELFEDGE_MODE	Visualization of a node's internal communication (depending on the respective measure value) using the diameter of a circle (only in **SIMPLE_NODE** mode). Valid values: - true (Show) - false (Do not show)
DEFAULT_EDGE_PIXEL_RANGE	Value range (floating point number) for connection weight in pixels (min, max) for dependent representation (all modes except **NONE_MODE**), for example, DEFAULT_EDGE_PIXEL_RANGE=1,3.5
DEFAULT_NODE_PIXEL_RANGE	Value range for diameter of circle in pixels (min, max) for representation depending on measure value (**DEFAULT_SELFEDGE_MODE=true**)
UNDER_WARNING_EDGE_COLOR	Connection color in RGB format for measure values that are better than the warning value specified (= planned value 1 for measure).
WARNING_EDGE_COLOR	Connection color in RGB format for measure values between the warning value (= planned value 1) and the alarm value (= planned value 2).
ALARM_EDGE_COLOR	Connection color in RGB format for measure values that are poorer than the alarm value (= planned value 2).

COLORS IN RGB FORMAT

The color values are specified as brightness steps of the three basic colors **red**, **green**, and **blue** (color triple). They are specified by three integers between 0 and 255 (256 steps). The first value gives the graduation of the red component, the second the graduation of the green component, and the third the graduation of the blue component. 0 is the lowest brightness step and 255 is the highest for a basic color value. Gray tones result from equal graduation, for example, 240,240,240.

Examples

Color	RGB format
Black	0,0,0
Dark gray	64,64,64
Deep red	255,0,0
Deep green	0,255,0

Color	RGB format
Deep blue	0,0,255
Brown	165,44,42
Gold	255,215,0
Light gray	240,240,240
White	255,255,255

6.1.3.17 RMI server

The information in this file configures the data exchange via the RMI protocol.

The **RMIObjectsPort** key specifies the port number used by the PPM client server. RMI objects created by the PPM client server are registered with this port number at the RMI registry.

The key **RMIObjectsPortAnalysisServer** specifies the port number that the analysis server uses for communication.

The **RMISocketFactory** key specifies the RMI data exchange type.

Value	Description
com.idsscheer.ppm.rmi.compress. ZCompressionSocketFactory	Compressed RMI data exchange (PPM default)
com.idsscheer.ppm.rmi. ZDefaultSocketFactory	RMI data exchange in plain text
No value	Native RMI communication **default**

If you have enabled SSL encryption (**UseSSL=true** key in the file **Registry_settings.properties**), the **RMISocketFactory** key is ignored and **ZSSLSocketFactory** is automatically used as the RMI SocketFactory.

6.1.3.18 Report

The multi-level keys distinguish between the different output formats of the PPM report component. Specific definition files are referenced for each output format based on the **report** directory in the client configuration directory. Comments are added to the keys in the file.

Key	Value	Description
editor.showInfoMessages	Boolean	Default value is FALSE After opening and saving an export definition, shows a notification dialog, if the value was set to TRUE.

Key	Value	Description
editor.execute.autosave		Default value is TRUE If the value was set to TRUE, the export definition is automatically saved before execution.
mnemonic_date_timezone		Default value is GMT+0:00 Use this setting to change the time zone for the **Date** field. This setting applies to exports only.

DEFINE EXPORT FORMATS

The export formats you can select in PPM are defined in the key **exportformats**. By default, the formats **XML**, **CSV**, and **CSVF** are available for export definitions and can be selected in the **Export properties** dialog.

Each format has a set of settings that need to be copied for a new format and adapted accordingly.

The key values must be specified in capital letters.

REPORT AUTOMATION OUTPUT DIRECTORY

The **reportautomation.result_directory** key indicates a directory (default: <installation directory>/ppm/server/bin/work/data_ppm/reportautomation) in which the report automation results are stored. An individual subdirectory is created for each client.

LOCALIZATION OF E-MAIL TEXTS

The **reports.resource_directory** key specifies a directory (default: <installation directory>\ppm\server\bin\work\data_ppm\config\<client>\report\resources) containing language-specific resource files for e-mail texts. The name pattern of these files is as follows: **report_resource_<language code>.xml**, for example, **report_resource_en.xml**.

In various context-specific XML structure elements, the texts are specified in the **PCDATA** box of the corresponding **resourceelement** XML element. The context of the corresponding text is specified in the **name** attribute of the **resourceelement** XML element.

You can efficiently create e-mail texts for languages that are not supported by copying the existing file **report_resource_en.xml** and rename it using the correct language code (for example, **report_resource_es.xml** for Spanish). All texts in the **PCDATA** box of this newly created resource file can then be translated to the relevant language in a text editor.

LOCALIZATION OF CPI TEXTS

You can edit interface texts and e-mail texts containing the output of particular values transferred by PPM (for example, planned values and measure values). To do so, specify particular XML child elements in the **cpiresource** XML element of the **report_resource.xml** file. In the **name** XML attribute of these child elements, the internal resource ID is specified which is then overwritten by the text specified in the **PCDATA** box:

```
<resourceelement name="<Resource ID>"><Text></resourceelement>
```

Example (extract from report_resource.xml)

```
<resource>
...
   <cpiresource>
...
      <resourceelement name="state">state</resourceelement>
...
```

EARLY ALERT CHECK

TOPIC: SUBJECT

Resource ID	ZEarlyAlertChecker.subject.STR
Variable {0}	Early alert dimension
Variable {1}	Process type
Text	Critical process instances concerning "{0}" ("{1}")

Resource ID	ZEarlyAlertChecker.message.exceeding_non.non_tv.warn_not_set.STR
Variable {0}	Process type group
Variable {1}	Early alert dimension
Text	Critical process instances were found for the early alert dimension(s) "{1}" under the process type group "{0}".

TOPIC: MESSAGE TEXTS

Resource ID	ZEarlyAlertChecker.message.clickHere.STR
Text	Click here:

Resource ID	ZEarlyAlertChecker.message.Information.STR
Text	Note: Problems occurred when connecting to the system. Therefore, some process instances might not have been included in the check.

Resource ID	ZEarlyAlertChecker.message.MessageDetails.STR
Text	For details, see the following message or contact your system administrator.

PLANNED VALUE CHECK

TOPIC: SUBJECT

Resource ID	ZPlannedValueAnalyticManager.subject.under_tv.STR
Variable {0}	Measure
Variable {1}	Process type
Text	Planned value below limit for measure "{0}" ("{1}")

Resource ID	ZPlannedValueAnalyticManager.subject.over_tv.STR
Variable {0}	Measure
Variable {1}	Process type
Text	Planned value exceeded for measure "{0}" ("{1}")

TOPIC: MESSAGE TEXTS

The resource IDs/texts described below use the following variables:

Variable {0}	Actual value
Variable {1}	Alarm value
Variable {2}	Warning value

Resource ID	ZPlannedValueAnalyticManager.message.exceeding_warn1.over_tv.STR
Text	Planned value 1 ({2}) has been exceeded. The actual value is {0}. Planned value 2 is {1}.

Resource ID	ZPlannedValueAnalyticManager.message.exceeding_warn1.under_tv.STR
Text	Planned value 1 ({2}) is below limit. The actual value is {0}. Planned value 2 is {1}.

Resource ID	ZPlannedValueAnalyticManager.message.exceeding_warn2.over_tv.warn_set.STR
Text	Planned value 2 ({1}) has been exceeded. The actual value is {0}. Planned value 1 is {2}.

Resource ID	ZPlannedValueAnalyticManager.message.exceeding_warn2.under_tv.warn_set.STR
Text	Planned value 2 ({1}) is below limit. The actual value is {0}. Planned value 1 is {2}.

Resource ID	ZPlannedValueAnalyticManager.message.exceeding_warn2.over_tv.warn_not_set.STR
Text	Planned value 2 ({1}) has been exceeded. The actual value is {0}. Planned value 1 has not been defined.

Resource ID	ZPlannedValueAnalyticManager.message.exceeding_warn2.under_tv.warn_not_set.STR
Text	Planned value 2 ({1}) is below limit. The actual value is {0}. Planned value 1 has not been defined.

TOPIC: TEXT FOR THE FILTER OF THE PLANNED VALUE DEFINITION

Resource ID	ZPlannedValueAnalyticManager.filter.STR
Text	The planned value definition refers to the following filters:

ALARM VALUE CHECK

TOPIC: SUBJECT

Resource ID	ZAlarmValueAnalyticManager.subject.under_tv.STR
Variable {0}	Early alert dimension
Variable {1}	Process type
Text	Alarm value below limit for measure "{0}" ("{1}")

Resource ID	ZAlarmValueAnalyticManager.subject.over_tv.STR
Variable {0}	Early alert dimension
Variable {1}	Process type
Text	Alarm value exceeded for measure "{0}" ("{1}")

TOPIC: MESSAGE TEXTS

Resource ID	ZAlarmValueAnalyticManager.message.exceeding_alarm.under_tv.warn_not_set.STR
Variable {0}	Alarm value

Text	Alarm value "{0}" is below limit.

Resource ID	ZAlarmValueAnalyticManager.message.exceeding_alarm.over_tv.warn_not_set.STR
Variable {0}	Alarm value
Text	Alarm value "{0}" is exceeded.

PROCESS MINING

TOPIC: SUBJECT

Resource ID	ZProcessMiningAnalyticManager.subject.STR
Text	Suspected deviations identified by Process mining

TOPIC: MESSAGE TEXTS

Resource ID	ZProcessMiningAnalyticManager.message.exceeding_non.non_tv.warn_not_set.STR
Text	Suspected deviations identified in Process mining.

OUTLIER ANALYSIS

TOPIC: SUBJECT

Resource ID	ZOutlierAnalyticManager.subject.STR
Variable {0}	Measure
Variable {1}	Process type
Text	Outliers identified for measure "{0}" ("{1}"

TOPIC: MESSAGE TEXTS

Resource ID	ZOutlierAnalyticManager.message.exceeding_non.over_tv.warn_not_set.STR
Variable {0}	Limit
Text	The outlier analysis identified outliers for the limit greater than "{0}".

Resource ID	ZOutlierAnalyticManager.message.exceeding_non.under_tv.warn_not_set.STR
Variable {0}	Limit

Text	The outlier analysis identified outliers for the limit smaller than "{0}".

6.1.3.19 Server

The data in this file is used for PPM server settings. The following table lists the most important setting options:

Key	Value	Description
PRINT_STACKTRACE_ ON_EXCEPTION	Switch	Is used for receiving additional internal details on error messages. TRUE activates the advanced output of error messages ("stack traces"). Default value: **FALSE**
ANALYSIS_SERVER_ CONNECT_RETRY_ INTERVAL	Figure	Number of milliseconds that the PPM server waits in case of a failed connection to the analysis server before starting a new connection attempt. Default value: **1500**
ANALYSIS_SERVER_ MAX_CONNECT_ RETRIES	Figure	Maximum number of connection attempts between PPM and analysis server before connection attempts are canceled. Default value: **120**
ENCODING	Text	Specifies the character set encoding of exported XML files. Valid values: **ISO-8859-1**, **UTF-8** This value is set in the client setup and must not be changed at a later time.
PREV_PERIOD. TOLERANCE _IN_PERCENT	Figure	Percentage tolerance within which comparison values are considered equal to the measure value. Used in EPC view to represent trends. Default value: **1**
SUBSERVER_RETRY_ WAIT_TIME	Figure	Number of milliseconds that the master waits in case of a failed connection to the subserver before starting a new connection attempt. Default value: **10000**
SUBSERVER_MAX_ CONNECT_RETRIES	Figure	Maximum number of connection attempts between master and subserver before connection attempts are canceled. Default value: **10**
SUBSERVER_RETRY_ WAIT_TIME	Figure	Waiting time in milliseconds after which the next attempt to connect to the sub-server is started Default value: **10000**

Key	Value	Description
SERVER_MODE	Text	Determines a server's operation mode. Valid values: **STANDALONE, SUBSERVER, MASTER** This value is set by client setup.
HTTPS_DISABLE_CERTIFICATE_AUTHORITY_VERIFICATION	Switch	**True** controls the check of the certification authority Default value: **false**. If the key is missing or has any other value the default case applies and the check will be performed strictly. If the check has been disabled the check of the certification authority will not take place for any of the https connections between the PPM server and other hosts. This does not only apply to connections with central user management. The RMI connection is unaffected by these settings. Further information on HTTPS support is available in the **PPM Operation Guide**.
HTTPS_DISABLE_CERTIFICATE_HOSTNAME_VERIFICATION	Switch	**True** turns off the check of the host name if the host name validation of the certificate during SSL communication is not to be performed. Default value: **false**. If the key is missing or has any other value the default case applies and the check will be performed strictly. This entry applies to all https connections. The RMI connection is unaffected by these settings. Further information on HTTPS support is available in the **PPM Operation Guide**.
LANGUAGE	Text	Language to be used for running the PPM server (in line with ISO 639-1. If the specified language is not supported by the client, the default language of the client is used) Default value: **„"** (not set)
QUERY_CONNECTION_POOL_SIZE	Figure	Size of the query connection pool for queries (> 0). Default value: **16**
COCKPIT_THREAD_COUNT	Figure	Number of threads used for calculating cockpit queries. (-1 for unlimited number of threads) Default value: **8**
USER_ADMIN_LIST_LIMIT	Figure	Maximum number of users displayed in user management when using LDAP (legacy or UMC) Default value: **100**

Key	Value	Description
PRIORITY_FREQUENCY_QUERY	Figure	Priority of frequency queries (integer >= 0). Smaller values have a higher priority. Default value: **8**
REPORT_TOKEN_TIMEOUT	Figure	Maximum runtime (in seconds) for report sessions (UMC). Default value: **604800** (60*60*24*7)
IMPORT_TOKEN_TIMEOUT	Figure	Maximum runtime (in seconds) for import and internal sessions (UMC). Default value: **604800** (60*60*24*7)
UMC_SESSION_REFRESH	Figure	Timeout extension of the UMC session. Multiplied by the value specified in UMC (default: 60 seconds). Default value: **15** (that is, 15 * 60 seconds = 15 minutes)
COMPARE_MASTER_SUBSERVER_CONFIGS	Switch	Indicates if the configuration is compared between master and sub-servers. This parameter is only relevant in the **MASTER** operation mode. Valid values are **TRUE** or **FALSE**. The default setting **TRUE** results in a comparison of the master configuration and the sub-server configuration. If the configurations are different, the sub-server will not connect with the master. The setting **FALSE** can lead to invalid and unpredictable behavior and should only be set if Software AG has instructed you to do so.
HTTP_PORT	Figure	Specifies the port number used by the load balancer to communicate with the PPM server.
WEB_SERVICE_SESSION_TIMEOUT	Figure	Validity time in milliseconds before a connection expires. If there is no communication with the server during this time, the user is logged out.
WEB_SERVICE_SESSION_TIMEOUT_RENEWAL	Figure	Time in milliseconds in which a logged in client sends a ping signal to the server to maintain the session.
WEB_SERVICE_MIN_THREADS	Figure	Minimum number of threads for Web service requests
WEB_SERVICE_THREAD_IDLE_TIMEOUT	Figure	Maximum duration of an inactive thread. If this time elapses with no activity, the thread is ended.

Key	Value	Description
WEB_SERVICE_REQUEST_TIMEOUT	Figure	Maximum wait time for a response from the server. If a request to the server takes longer than the time specified in this value, the server sends a timeout to the client so that the client can request the result again. This value must be lower than the abort time between the upstream load balancers and the client server.

During the master server client setup, all sub-servers to be used by the master are specified. This information is stored in the following entries (X is a placeholder for the number of a sub-server, value range 1-64):

Key	Value	Description
SUBSERVER.X.URL	URL	Registry URL to address the sub-server
SUBSERVER.X.CLIENT	Text	Name of the client
SUBSERVER.X.USER	Text	Name of the PPM user whose ID is used by the master server to connect to the sub-server
SUBSERVER.X.PASSWORD	Text	Encrypted password of the PPM user
PIKI_SUBSERVER	Figure	Determines the sub-server (value corresponds to the number of the sub-server) that contains the measure series for data input of process instance-independent measures in the PPM front-end.

It is recommended to use a system administrator user for the communication between master server and sub-servers. This ensures that the master server can access the sub-servers with full system privileges.

The data for process instance-independent measure series should always be imported on the sub-server specified in PIKI_SUBSERVER.

Warning

Distributing data of a process instance-independent measure series on several sub-servers can lead to invalid results.

6.1.3.20 Server_Log

The data in this file is used for system messages settings of the PPM server.
Detailed information is available in the **PPM Operation Guide**.

6.1.3.21 Sysmon

The following settings are configured for the system monitor client.

Key	Value	Description
GENERATE_EVENT_XML	Switch	Is used for generating event files for the system monitor client during PPM import. TRUE activates the generation. Default value: FALSE
PPM_EVENT_DIR	Text	Location where the generated event files are saved. Default value: <installation directory>/ppm/ server/bin/work/data_ppm/custom/sysmon/data/SysmonDat a
CHECK_DB_STATUS	Switch	Querying the DB status at specific measurement points. TRUE can negatively affect performance. Default value: FALSE
ATTR_AT_PPM_CUSTOMER	Text	User-defined attributes can be specified with the **ATTR_** prefix. They are copied to each event. These attributes must be maintained in Customizing and mappings need to be created for them.

6.1.3.22 Templates

The following settings are specified for reports:

Partial key	Meaning
BaseDirectory	Directory containing the respective definition files (access by the file system). The system user executing the PPM software requires **Full access** mode for this directory.
BaseURL	Specifies the URL access by HTTP for Management views and by FILE for reports to the directory indicated under BaseDirectory.
Filter	File extension of report definition files.
QueryAPI.Context	Name of the query API context for the client
QueryApi.PPMUIMode	Jump target for PPM URLs created by the query API. Valid values: APPLET, WEBSTART. **APPLET** specifies **index.html** and **WEBSTART** specifies **ppm_<client>.jnlp** Default value: WEBSTART

Partial key	Meaning
Optimize.UseSSO	Single sign-on for Optimize. The only valid value: SAML2
Optimize.BaseURL	Jump target for Optimize
Mashzone.BaseURL	Jump target for MashZone NextGen
Mashzone.UseSSO	Single sign-on for MashZone NextGen. The only valid value: SAML

Base directory and base URL are needed by the templates, that is, the registered reports.

The base URL for reports needs to be specified for completeness sake only.

6.1.3.23 MT_Export

The following settings are configured for exporting Minitab graphics:

Partial key	Meaning
MT_graphic_heigth	Specifies the height of a graphic. The default value is 480.
MT_graphic_width	Specifies the width of a graphic. The default value is 760.