How-to Guide
SAP NetWeaver

How To... Configure CCMS Monitoring for MDM 7.1 ABAP API Tracing

May 2009

THE BEST-RUN BUSINESSES RUN SAP

Table of Contents

1 MDM 7.1 - CCMS Monitoring for ABAP API Applications

MDM ABAP API CCMS monitoring is based on the:

- **Central Monitoring System (CEN)**

  CCMS monitoring is based on the ABAP stack of a Web AS (Web Application Server). The CCMS application collects the monitoring information for all connected systems and applications in this system.

  This central monitoring system is called ***CEN***.
- **ABAP system connecting MDM via ABAP API (called APISys)**

  MDM delivers an API technology that allows you to connect MDM repositories running in an MDM Server environment from any ABAP system in which the MDM ABAP PI is installed. This ABAP system can be for example an R/3 system or a CRM system.
- **MDM Server Environment**

  The MDM repositories run on an MDM Server (Win OS or MDM UNIX platform). The ABAP API can connect to all repositories loaded on a running MDM Server in the MDM system landscape.

Prerequisite for this documentation:
Before starting installation of the MDM 7.1 monitoring environment:

- Read the *MDM 7.1 Solution Operations Guide*.
- Make sure that you have sufficient authorization in the central monitoring system CEN.
- Make sure that you have administration rights to perform all necessary installation steps on the local monitored ABAP system.

Installation of the MDM 7.1 CCMS monitoring environment includes:

- Installation of a central monitoring system (CEN) based on a WAS system (6.40 or higher recommended). Installation of the CEN is not included in this documentation.
- Installation of the local agents and tools running on the MDM Server machine for MDM Server monitoring
- Installation of the standard MDM CCMS monitoring template

Configuration of the MDM 7.1 CCMS monitoring environment includes:

- Registration of the monitored system components (ABAP API systems and MDM Server systems) on the CEN
- Definition of the MDM monitoring tree in the CEN

This How To guide differentiates between the following **types of monitoring**:

- **MDM Server**: Monitor the MDM Server components (C++ applications running on a Windows PC or UNIX machine)
- **ABAP API**: Monitor the MDM ABAP API running on any ABAP system on which the MDM add-on is installed and run both an application and a kernel trace for the ABAP API environment.
  Note: Kernel traces are only available for MDM ABAP API providers < 7.1.

  The ABAP API part is explained in detail in this guide; the MDM Server part can be found in a separate MDM How To Guide.

All **examples** in this guide are based on the following architecture: an MDM Server running on a PC with Windows OS together with the other MDM components. A dedicated ABAP system plays the role of the CEN (**CEN**) and an ABAP system uses the ABAP API (**API**) to communicate with the MDM Server and its repositories.

2 MDM ABAP API Monitoring Overview

The following image displays a typical MDM monitoring landscape. The CEN receives all monitoring data from remote CCMS agents, displays them in a monitoring tree, and allows activities based on the monitoring data, such as Alert reaction or aggregated historical storage of monitoring data.

Error! Objects cannot be created from editing field codes.
The CEN is the common system centrally containing all MDM monitoring data, independent of the source of the monitoring data:

MDM Server monitoring is based on the SAP CCMS architecture. A CCMS agent is installed on the MDM Server and a library that allows communication between the MDM Server and CCMS is plugged in.
The CCMS agent on the monitored system (for example PC on which the MDM Server is installed) collects monitoring information, such as the availability of the MDM Server, MDM log files, or OS Collector data about the system status. This data is provided via RFC to the central monitoring system on which the CCMS agent is registered. The CCMS agent can be registered on more than one monitoring system, but one of these monitoring systems is the central monitoring system (CEN).

ABAP API monitoring is also based on the SAP CCMS architecture. In the API system, you must start a monitoring data supplier that controls the monitoring data transfer from the API system to the CEN. The data is transferred using the CCMS infrastructure,
All monitoring data is provided to the local CCMS database in the API system and then transferred to the central CEN. All supportability and operability information can thus be displayed and evaluated at a central location.

The following data from an API system can be monitored in the CEN:

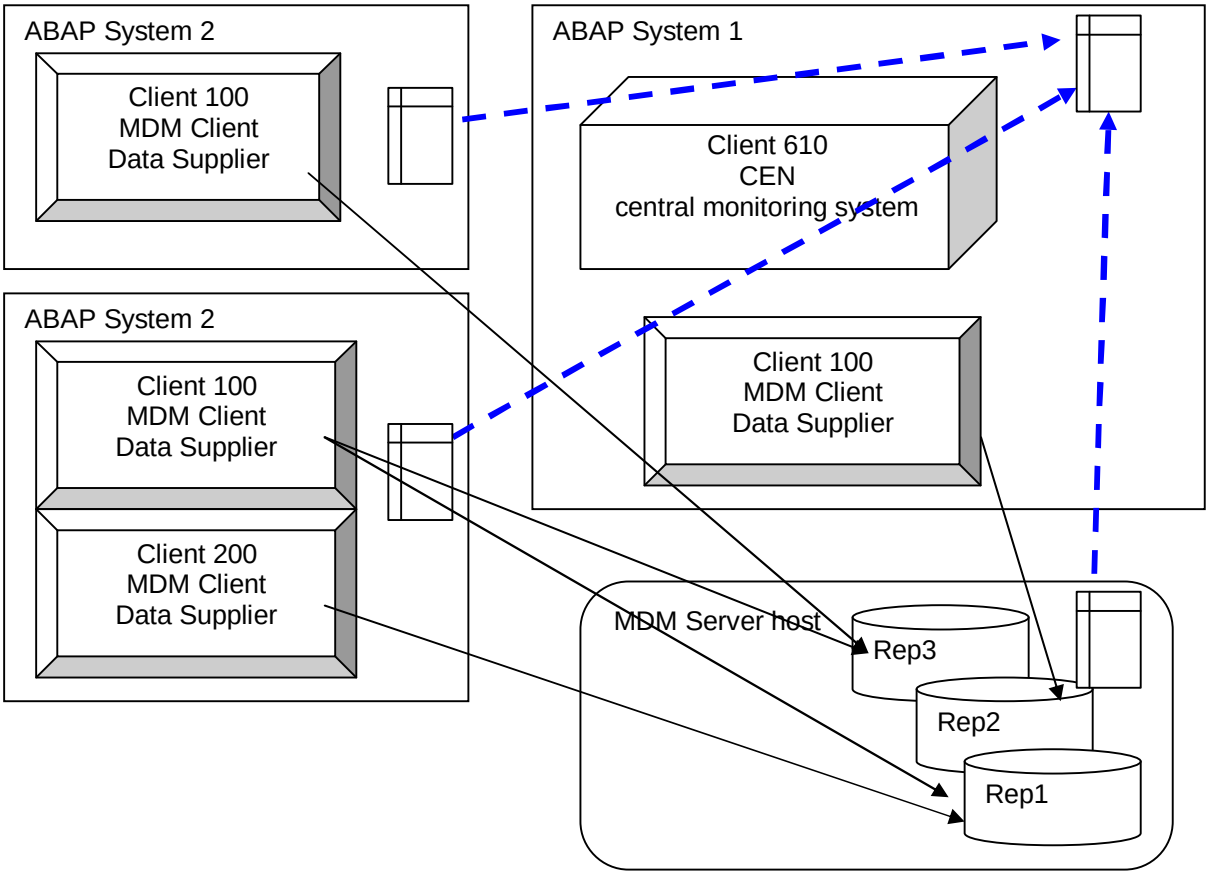
- API communication channel availability
  CCMS (RZ20) provides an overview of the communication channels defined between the API system and the MDM Server.
  The monitoring tree displays the availability information of the channels together with some performance attributes, such as the response time from ABAP to MDM, making it easier to detect performance leaks between ABAP and MDM.

- Advanced Logging and Tracing
  CCMS also provides the ABAP API Control Center function, which allows you to activate and deactivate application traces and kernel traces for the ABAP API programs running in the API system.
  This function is described in detail in How To Guide *MDM ABAP API How To Guide CCMS – Advanced Logging and Tracing Using CCMS.*

This guide covers all configuration steps needed to set up CCMS monitoring on an MDM ABAP API system and explains the use of communication channel availability monitoring.

3 CCMS in ABAP Landscape – MDM-Specific Issues

MDM ABAP API monitoring handles some specific features of the CCMS system architecture and the CCMS data transport. This chapter provides a basic knowledge of the setup steps to be performed.



The ABAP CCMS architecture works as follows:

- CCMS uses exactly 1 Shared Memory segment to transfer data for each ABAP system.
- The CCMS monitoring data is transferred between systems using RFC connections. The CCMS capabilities together with Shared Memory are available in each ABAP system and can be provided for non-ABAP systems (e.g. the MDM host) using CCMS agents.
- If more than one ABAP client in an ABAP system sends monitoring data to the CCMS, they use one common Shared Memory. The fact that client-specific data is to be displayed therefore must be highlighted in the monitor tree itself.
- If analysis methods have been started, e.g. to perform local application calls on the remote CCMS client, the client data must be evaluated to ensure that the application is called in the correct client using RFC.
- If a client provides a monitoring tree, the tree has to be signed with a client-specific signature.
- If a remote monitoring tree element is defined in CEN with an analysis method, this method is called on the remote ABAP system/client.

The MDM ABAP API can be used from any ABAP client on any ABAP system in the ABAP landscape. This fact results in some configuration effort for the different ABAP API monitoring types:

- MDM Availability Monitoring
  The Availability Data Supplier has to be started on each client on which the ABAP API is active.
  The monitoring tree elements are signed with their system name and client.
- Advanced Tracing
  Core is the “control center” application in which CCMS analysis methods are used to start different applications on the remote CCMS system that could contain different clients.
  In SP04 you can set up different analysis methods on the different clients; these are called later by the CCMS methods.

The CEN communicates with one dedicated client on the monitored system. If additional clients are running ABAP API and need to be monitored, the CCMS on that dedicated client needs to call additional client-specific CCMS methods for the client-specific data collectors.

The following chapters describe the steps for ABAP API CCMS configuration.

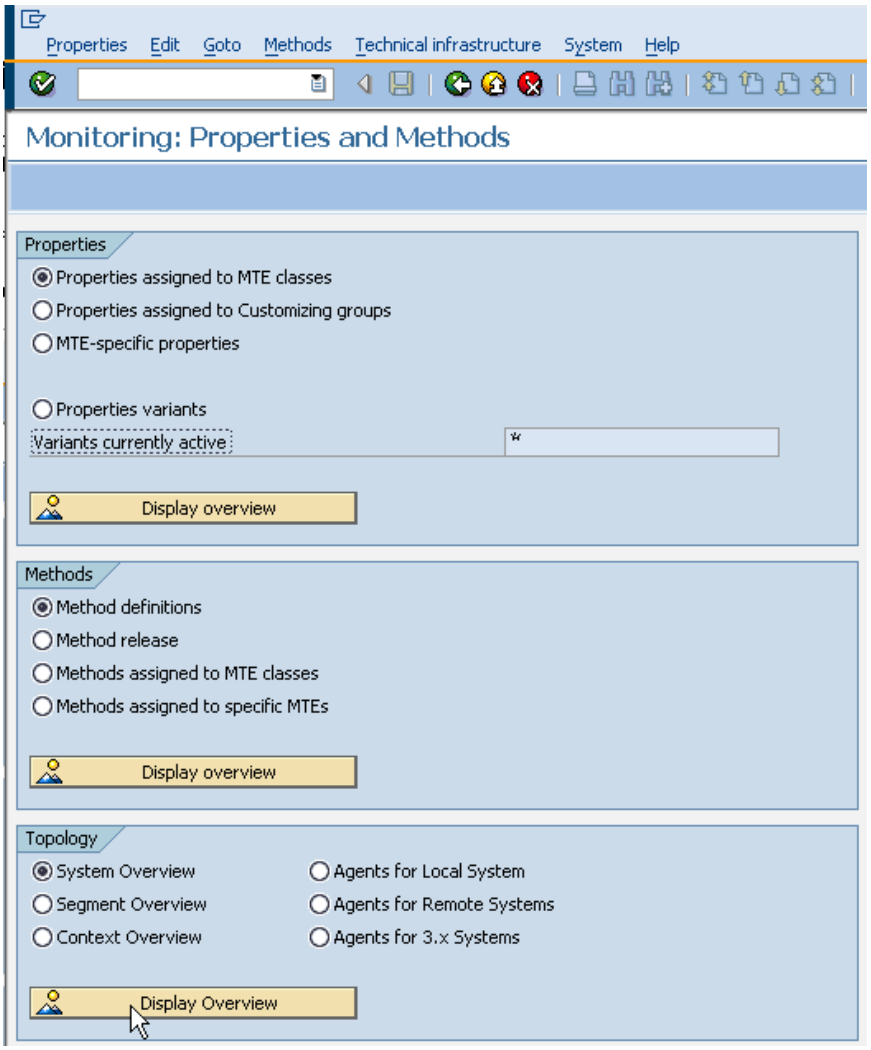
4 Create Connectivity Between CCMS Client and CEN

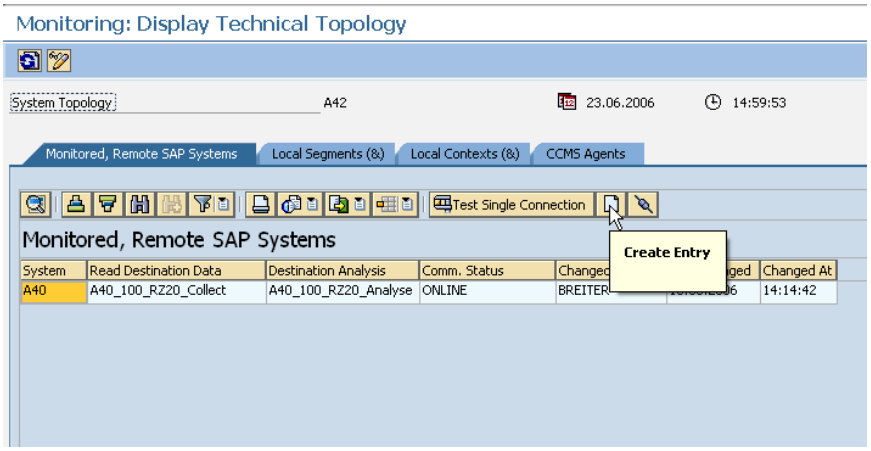
This chapter assumes that the ABAP system on which the MDM ABAP API is running is different from the CEN. A Solution Manager often plays the role of the central monitoring system CEN.
Both systems are up and running.
Two different users are needed to create RFC connections for the remote system.
The first user is responsible for performing analysis methods; the second one is responsible for collecting the monitoring data. The CCMS infrastructure allows you to create the RFC destinations directly from the CCMS.

1. Start the CEN and call transaction RZ21 to define the monitoring infrastructure.

   Mark “System Overview” and click “Display Overview” in the Topology section.

   On tab “Monitored, Remote SAP Systems”, create a new communication channel by clicking “Create Entry”.

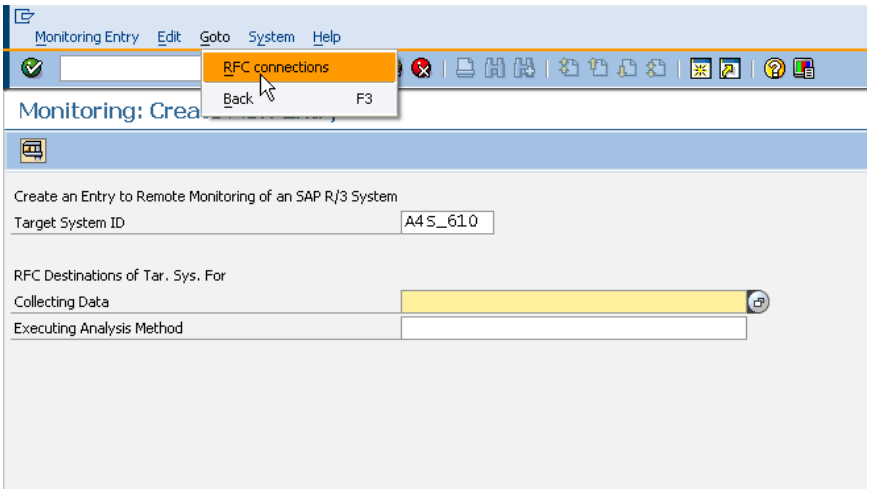




2. Enter the Target System ID.

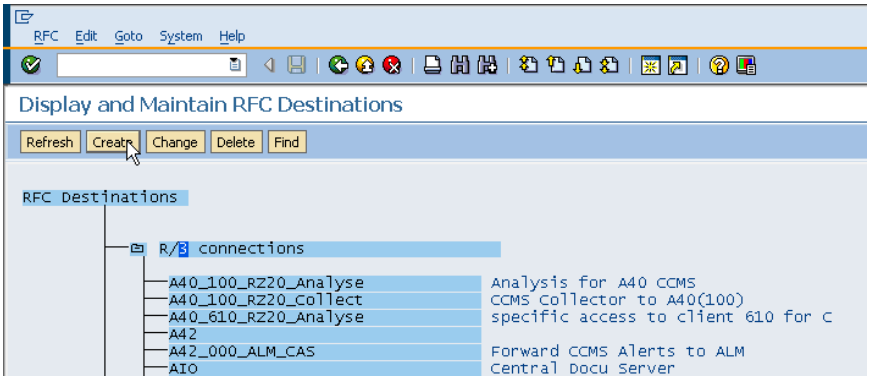
   Suggestion: Use the system and client if you want to connect different clients as described in chapter 3 for Advanced Tracing.

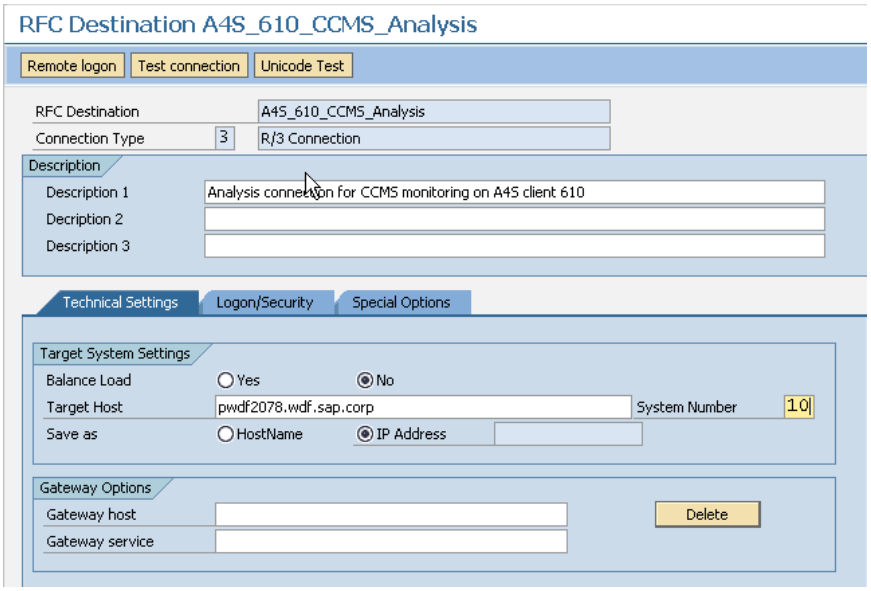
   If you have not defined RFC destinations for Analysis and Collection, choose “Goto” → “RFC connections” to create two RFC destinations.



3. Create R/3 connections (type 3) with the following data:

   We recommend the following names:
   <SYS>_<CLNT>_CCMS_Analyse
   <SYS>_<CLNT>_CCMS_Collect





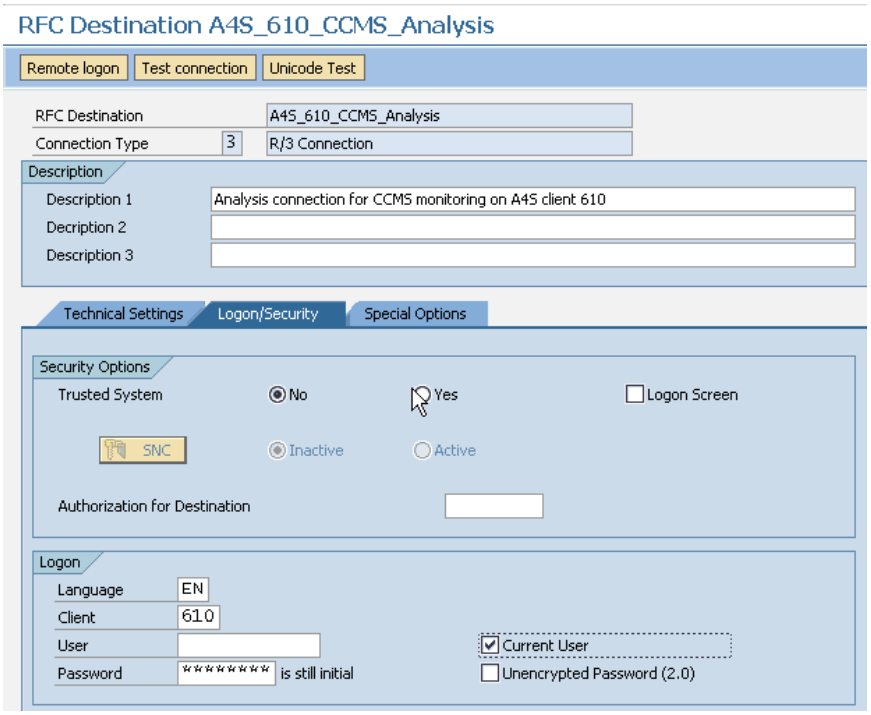
4. RFC destination
<SYS>_<CLNT>_CCMS_Analyse

Mark “Current User”.

Choose “Test Connection” and “Remote logon” to verify the RFC connection.

<SYS>_<CLNT>_CCMS_Collect

To avoid displaying the logon mask when starting the monitor, enter an existing user and password. The user needs CCMS authorization.

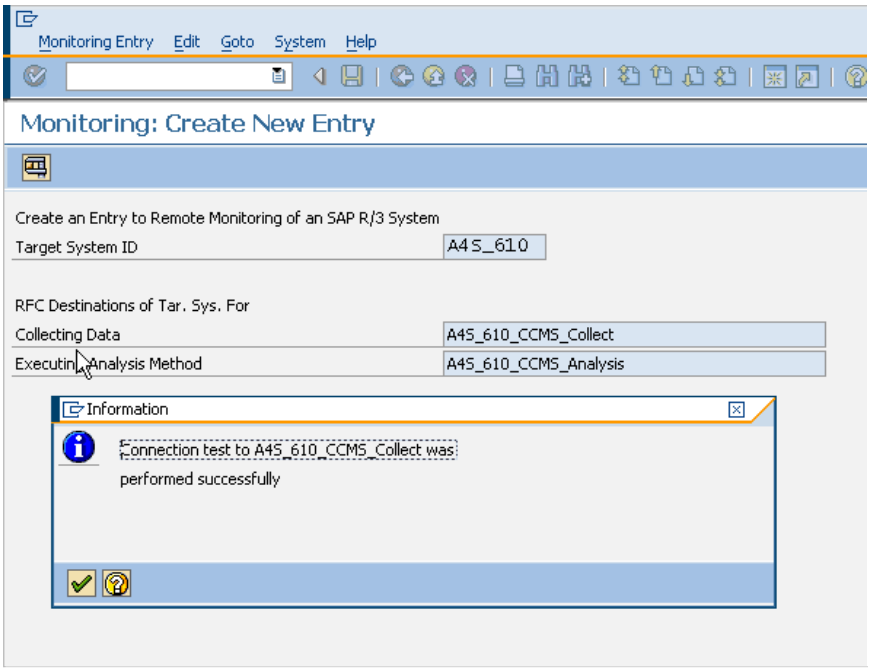


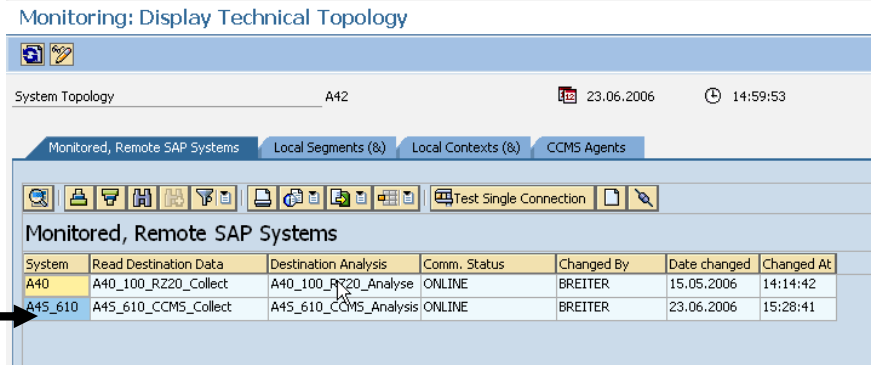
5. After creating the two RFC destinations, enter them in the “Monitoring: Create New Entry” screen.

Test the RFC connection and save the CCMS connection.

If you go back (F3), you should see the new connection to the new remote system.

Choose “Test Single Connection” to test if the connection was set up successfully.





As mentioned in chapter 3, you need to perform this connectivity step for

- each ABAP system that is acting as MDM ABAP API channel and that has to be monitored in CEN
- each client in a remote ABAP system if these clients use the MDM ABAP API in parallel and you want to perform advanced monitoring and tracing.

5 Monitored System (API) – MDM_TECH Package: Delivery Content

The tools, methods and software necessary for monitoring and tracing ABAP API usage are shipped with the SAP MDM ABAP API. The complete content is delivered with package MDM_TECH (for MDM 7.1 this is MDM_TECH_710).

The delivery contains the following content for setting up and using MDM ABAP API CCMS monitoring:

1. MDM-specific CCMS methods
2. MDM-specific CCMS monitoring template
3. MDM data suppliers

5.1 MDM-Specific CCMS Monitoring Template

Package MDM_TECH contains an MDM-specific CCMS monitor template. Instead of searching for all MDM monitoring data in different CCMS-specific monitors, such as “SAP CCMS Technical Expert Monitors / All Monitoring Contexts”, you can check all MDM-relevant monitoring data in ONE tree to find all monitoring contexts from all monitored systems.

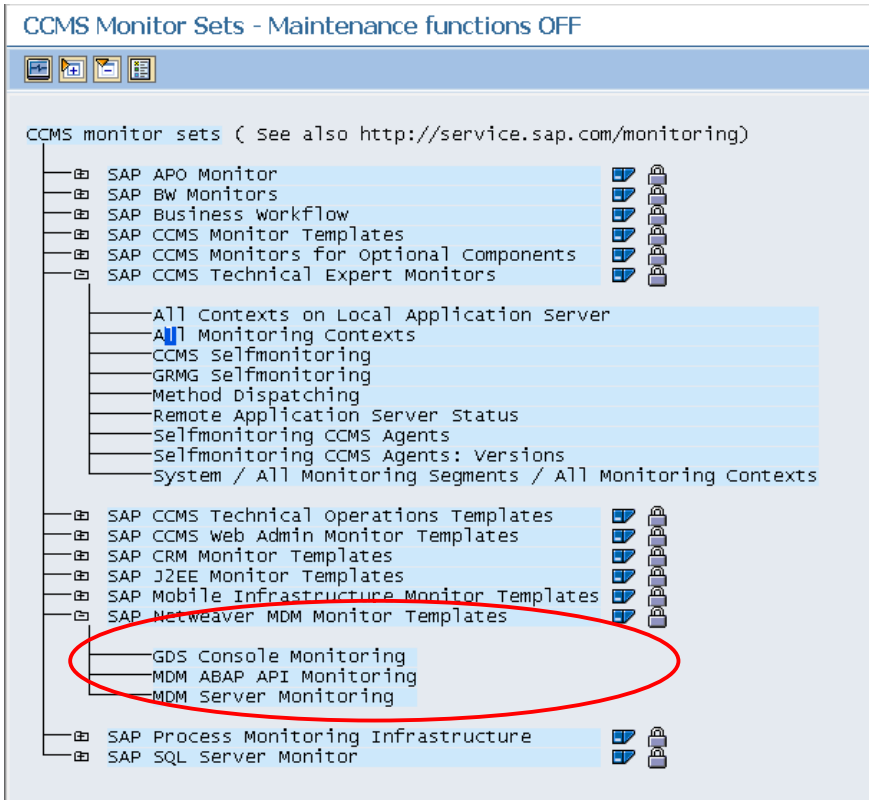
You can find descriptions of all CCMS standard monitor templates in the standard documentation area for CCMS.

All MDM-specific monitoring data is combined in a single template. This template will be enhanced with new functions in the future.
The MDM monitor template defines the “CCMS entry points”, which, technically speaking, are MTEs (monitoring tree elements). Rule nodes are used to define and collect the MDM monitoring data.
The MDM-specific MTEs are delivered by the MDM Data Collectors that run on each monitored MDM system.

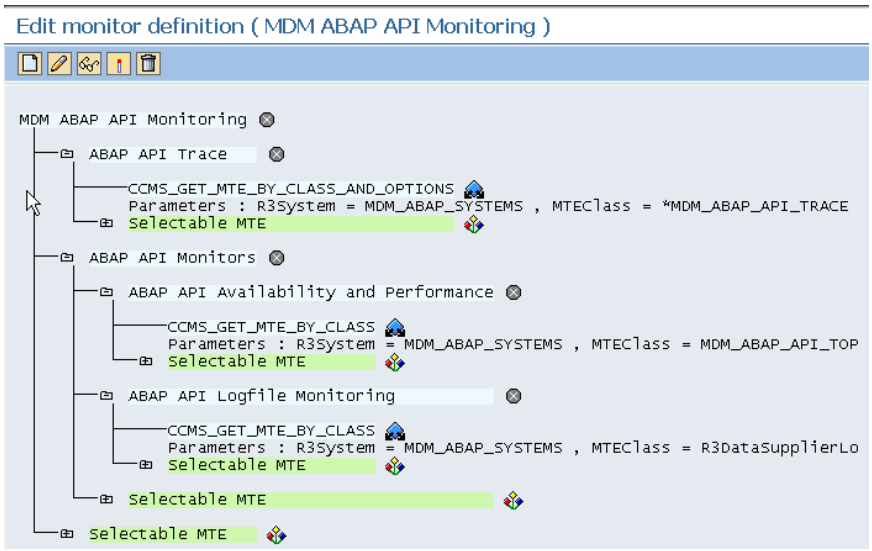
Modifying the delivered MDM monitor template:

Since you may not change delivered SAP objects, you should copy the MDM monitoring template to a customer-specific tree and make your modifications there.

1. Start transaction RZ20 in the central monitoring system (CEN). A number of predefined templates is displayed. One of these is the MDM-specific template (see screenshot).



2. Example for the rule-based monitor template definition in CCMS: MDM ABAP API Monitoring



5.2 MDM-Specific CCMS Methods - Overview

The following three CCMS methods can be called from the CCMS interface:

- **Data Collector** methods
  These methods are used to collect the data in the monitored system clients and to provide this data to the central monitor.
- **Analysis** methods
  When monitoring data is displayed in the monitoring tree, a double-click calls the Analysis method. The idea behind this is to jump to the monitored system for the parameter and to check / analyze the environment on the monitored system.
- **Auto-reaction** methods
  These methods allow you to define customer-specific reactions to specific status changes in the central monitor. The most frequent use is to send an alert notification when a specific threshold in a monitored parameter is violated, for example send an e-mail to a system manager if the CPU is overloaded.

These methods can be used out-of-the-box, but have template character. Different behaviors can be defined in the method definition, for example the server on which the method should run or how it can be called (automatically or manually). For details about the CCMS method definitions, see the standard CCMS documentation.

The following methods are delivered with MDM 7.1 MDM_TECH. Some methods are defined for client-specific call if you connected different clients for monitoring.

5.2.1 MDM ABAP API Logging and Tracing Methods

Note: Kernel traces are only available for MDM ABAP API providers < 7.1.

These methods are used in the delivered MDM-specific CCMS monitor template “SAP Netweaver MDM Monitor Templates” in subtree “MDM ABAP API Monitoring → ABAP API Trace”. Without the _sysclnt extension, the method works automatically by calling the monitored system from the monitoring system (CEN), whereas if you use different ABAP API clients on the same system, you need to adapt the _sysclnt methods using explicit RFC destinations.

MDM_ABAP_API_APPL_TRACE_APLG_CONF	Analysis	Jumps to the standard configuration for the SAP application log on the monitored system
MDM_ABAP_API_APPL_TRACE_APLG_CONF_sysclt	Analysis	--“-- but defined for client-specific call
MDM_ABAP_API_APPL_TRACE_CONF_ANA	Analysis	Jump to the MDM-specific logging/tracing configuration
MDM_ABAP_API_APPL_TRACE_CONF_ANA_sysclt	Analysis	--“-- but defined for client-specific call
MDM_ABAP_API_APPL_TRACE_DEL_ANA	Analysis	Delete expired logs on the monitored system
MDM_ABAP_API_APPL_TRACE_DEL_ANA_sysclt	Analysis	--“-- but defined for client-specific call
MDM_ABAP_API_APPL_TRACE_DISP_ANA	Analysis	Display MDM ABAP API application logs on the monitored system
MDM_ABAP_API_APPL_TRACE_DISP_ANA_sysclnt	Analysis	--“-- but defined for client-specific call
MDM_ABAP_API_KRNL_TRACE_KF_ANA	Analysis	Control Kernel Function Tracing on monitored system and display Kernel Function trace. The method also allows you to select different application servers. **Only available for MDM ABAP API providers < 7.1.**
MDM_ABAP_API_KRNL_TRACE_KF_ANA_sysclnt	Analysis	--“-- but defined for client-specific call. **Only available for MDM ABAP API providers < 7.1.**
MDM_ABAP_API_KRNL_TRACE_WP_ANA	Analysis	Work process trace on monitored system. The method also allows you to select different application servers. **Only available for MDM ABAP API providers < 7.1.**
MDM_ABAP_API_KRNL_TRACE_WP_ANA_sysclnt	Analysis	--“-- but defined for client-specific call. **Only available for MDM ABAP API providers < 7.1.**

The MDM ABAP API tracing control center allows you to jump to the specified client on the remote ABAP system. The method attached to the node as analysis method therefore needs to be client-specific.

The CCMS on CEN first connects to the Shared Segment on the monitored system. The specified method on the remote system is then called → in this case you need to create the client-specific method on the monitored system and enter the method on the monitoring tree on the CEN.

The definition method is described with an example method MDM_ABAP_API_APPL_TRACE_APLG_CONF_sysclt:

Perform the method definition and attachment for these 6 methods:
MDM_ABAP_API_APPL_TRACE_APLG_CONF_sysclt
MDM_ABAP_API_APPL_TRACE_CONF_ANA_sysclt
MDM_ABAP_API_APPL_TRACE_DEL_ANA_sysclt
MDM_ABAP_API_APPL_TRACE_DISP_ANA_sysclnt

Only available for MDM ABAP API providers < 7.1:
MDM_ABAP_API_KRNL_TRACE_KF_ANA_sysclnt
MDM_ABAP_API_KRNL_TRACE_WP_ANA_sysclnt

Attach the methods to the monitoring tree for the following 6 tree elements:

Application Trace
  ABAP API Application Trace – Configuration
  ABAP API Application Trace - Display
  ABAP API Application Trace - Delete
  ABAP API Appl Trace - Appl Log Config

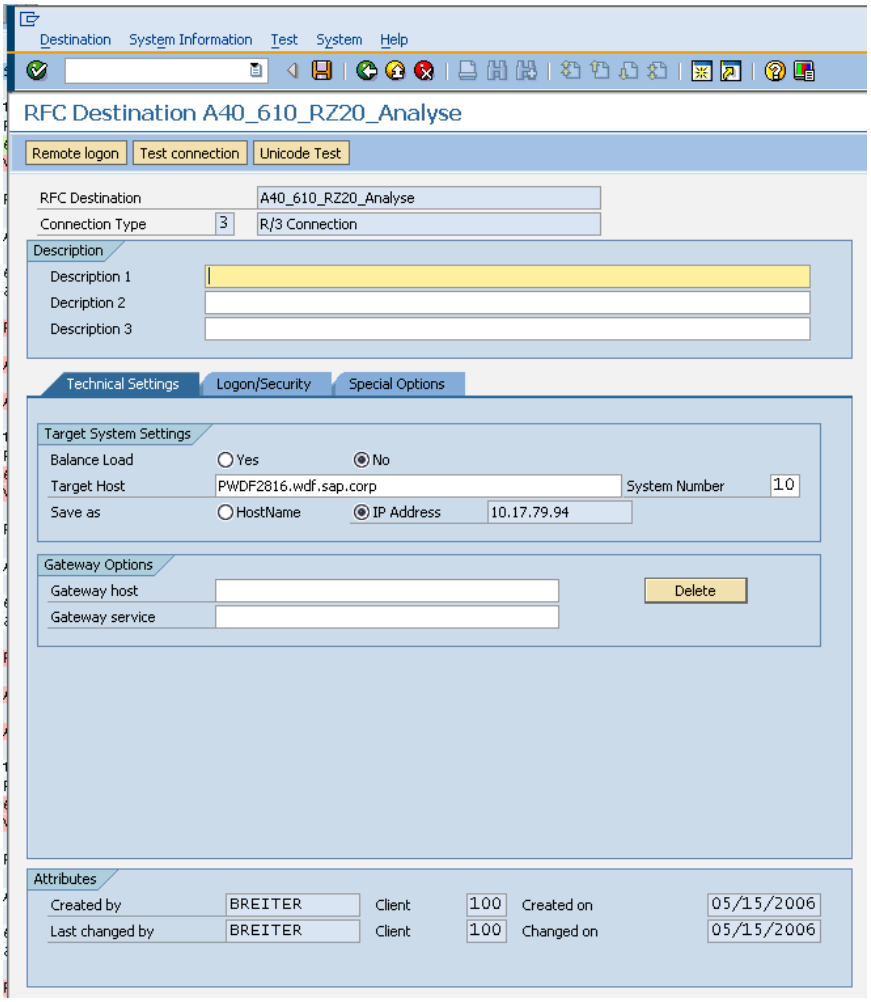
Only available for MDM ABAP API providers < 7.1:
Kernel Trace
  ABAP API Kernel Trace - Work Processes
  ABAP API Kernel Trace - Kernel Functions

1. Monitored system:

   Check if there is an RFC destination to the remote client (in this example CEN A42 connects to client 100 and 610 in A40: → In A4S you need two RFC destinations (A4S_100_Analysis and A4S_610_Analysis).

   The monitored system is defined as connected system in the CEN (RZ21, Monitored, Remote SAP Systems).

   In the demo example, since client 100 is connected, you also need an RFC destination from client 100 to 610 to call the tracing transactions on client 610.

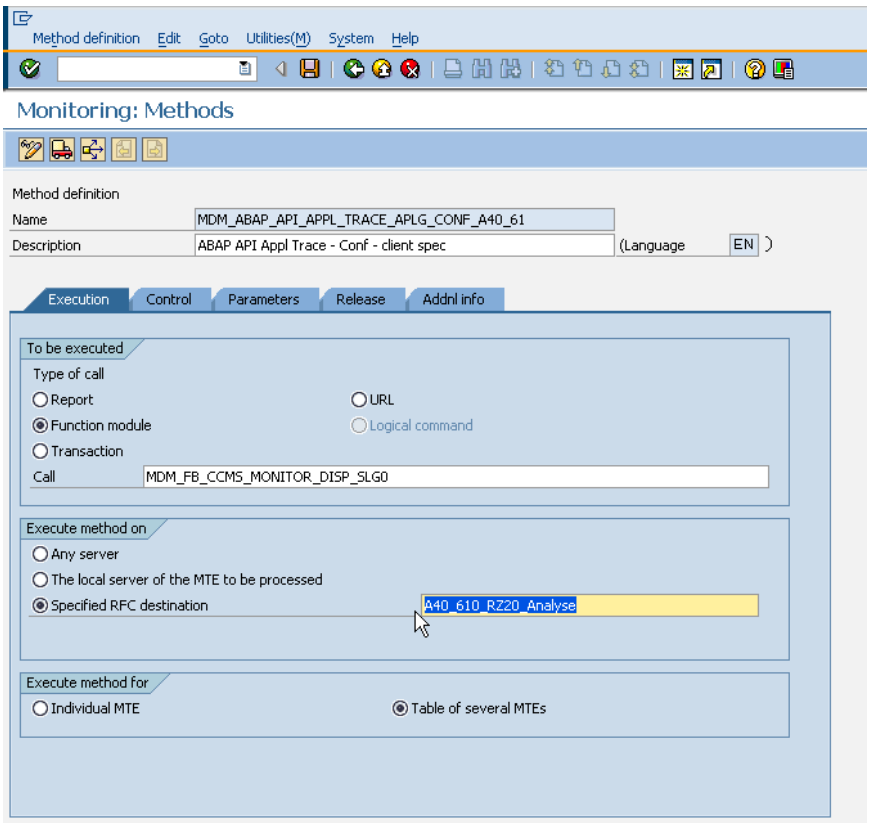


2. Monitored System:

   Start transaction RZ21 in the monitored system to create the new (RFC-specific) method.

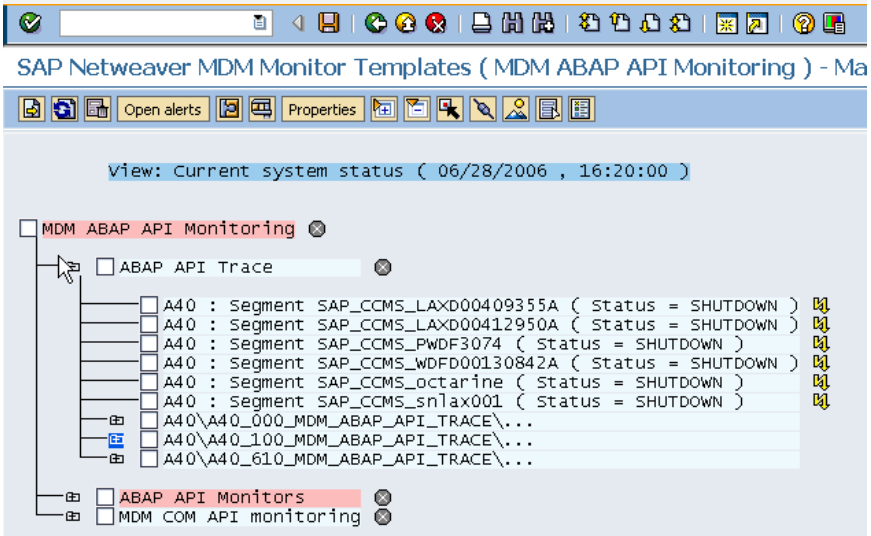
   Mark checkbox “Specified RFC destination” and enter the defined RFC destination from the connected CCMS client on the monitored system to the destination client.

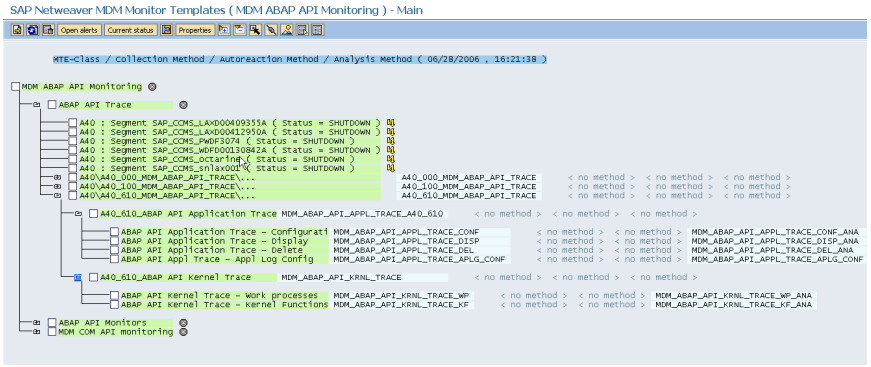
   Also perform these steps with the other methods.



3. Monitored System:

   Start transaction RZ20 to attach the client-specific method to the correct subtree (in our case A40\A40_610_MDM_ABAP_API_TRACE\...). Change to *Method Allocation View* to see the allocated analysis methods on the different tree items.





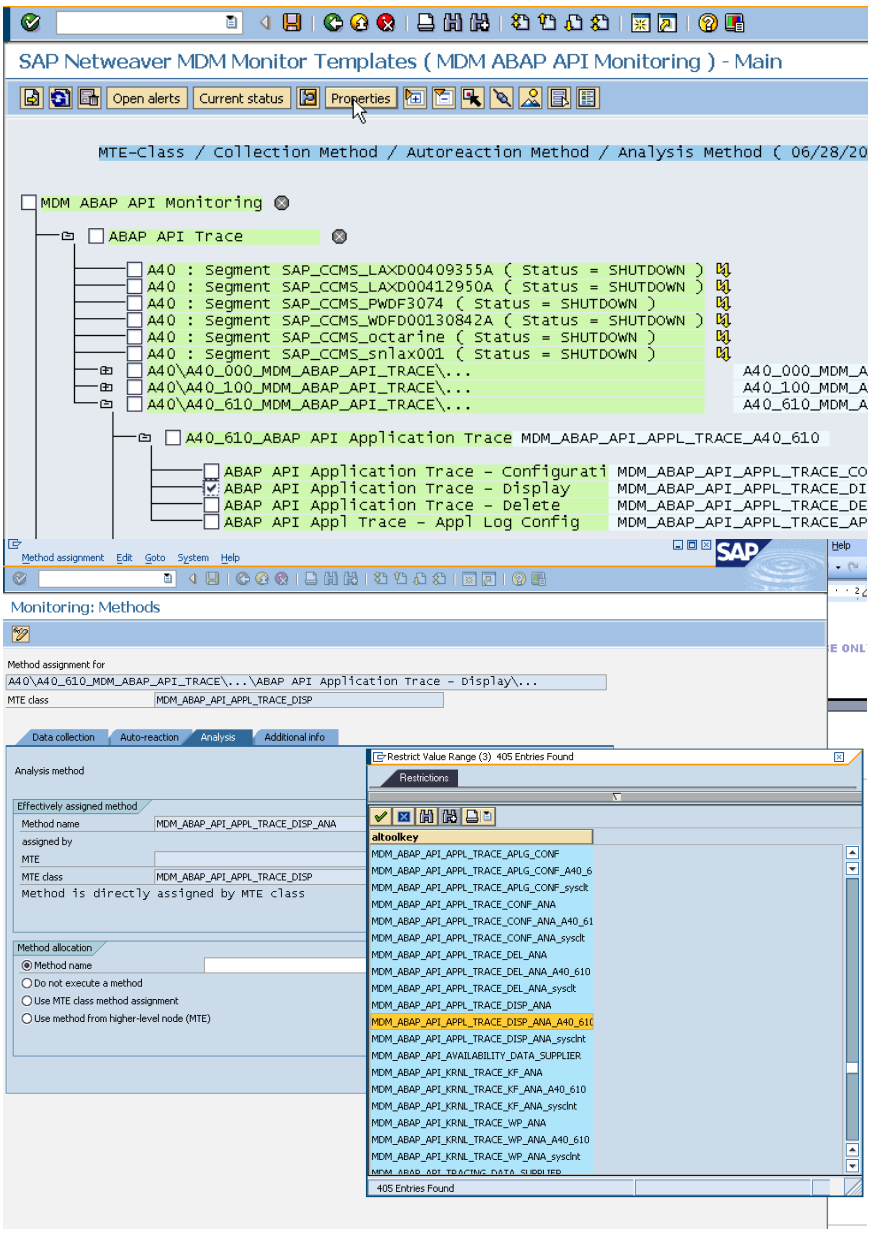
4. Select all 6 tree elements mentioned above.

   This is shown here for tree element “ABAP API Application Trace – Display”.

   Select the tree element and choose Properties.

   Go to tab Methods, click “Methods assignment”, and change the method as shown in the screenshot.

   Go to tab Analysis and enter the client-specific method (using F4 value help). Save the method assignment.



5.2.2 MDM ABAP API Data Suppliers

These methods are basic and need to be started once per monitored client. They are defined so that they are automatically started at system startup (method definition detail: *Execute method immediately after monitoring segment start*)).
These methods automatically create the monitoring subtree for API availability monitoring for each monitored system.
After being started once, the CCMS infrastructure makes sure that the methods are called periodically to collect the monitoring data from all monitored systems (for details see the CCMS standard documentation).

MDM_ABAP_API_TRACING_DATA_SUPPLIER	Data Collector	Defines the ABAP API Advanced Logging and Tracing Tree

Calling Availability Data Collector for Different ABAP API Clients

The tracing data collector only needs to be called once in the monitored system (client- independent), whereas the data collector for availability needs to be called as a separate method for each client.
The method without the _sysclnt extension works automatically by calling the monitored system from the monitoring system (CEN), whereas if you use different ABAP API clients on the same system, you need to adapt the _sysclnt methods using explicit RFC destinations.

The Availability Data Collector needs to be called once to create the tree and the tree elements, but availability also has to be checked periodically, and the data collector needs to be restarted periodically in order to do this. The period is defined in the properties of the performance parameters for the availability and can be changed by the user.
The CCMS then schedules a restart of the planned data collector after a specific time; this is done for each client-specific collector method.

MDM_ABAP_API_AVAILABILITY_DATA_SUPPLIER	Data Collector	Defines the ABAP API Availability Monitoring Tree
MDM_ABAP_API_AVAILABILITY_DATA_SUPPLIER_sysclt	Data Collector	--"-- but defined for client-specific call

Both data suppliers allow you to define parameters that specify the client to be checked for ABAP API connections and the trace level of the CCMS data suppler self-monitoring, which is written by the data suppliers and can be evaluated in the MDM monitoring tree.

These parameters are:

Parameter	Description	Value
TRACE_LEVEL (0,1 or 2)	Level of information written to the Self-Mon log for the data supplier function. ZERO does not write any information, STANDARD is restricted to a few control messages, while ENHANCED writes the result of each CCMS call to the protocol. The screenshot at the end of the chapter shows how to access the protocol.	2: ENHANCED 1: STANDARD (default) 0: ZERO

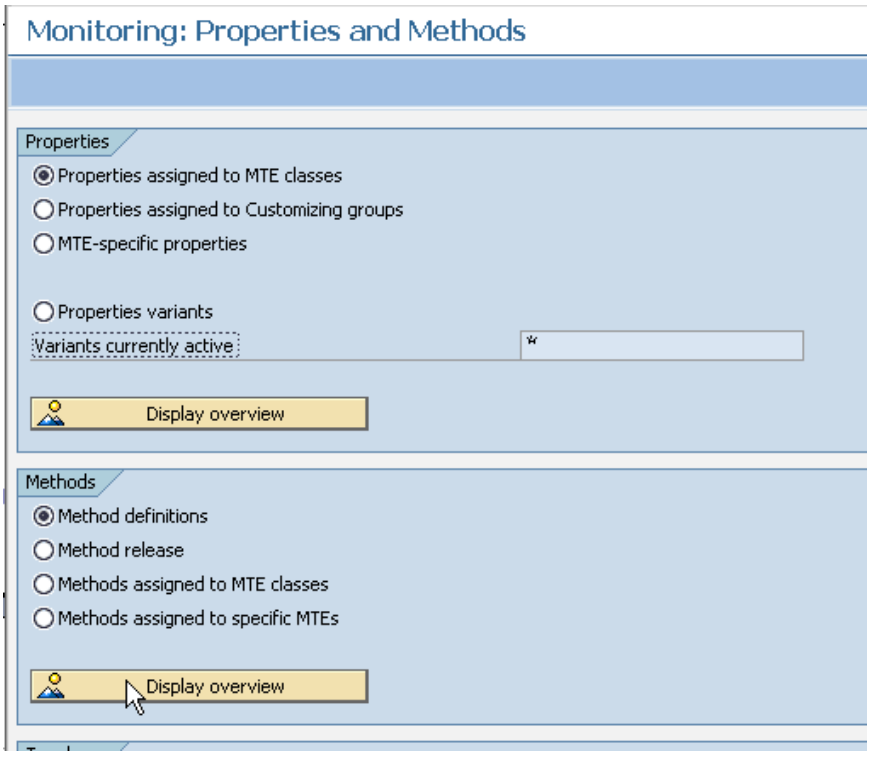
Set up the data supplier methods including parameter definitions. The example uses a client-specific call.

1. Monitored system:

   Start transaction RZ21 to define the CCMS infrastructure.

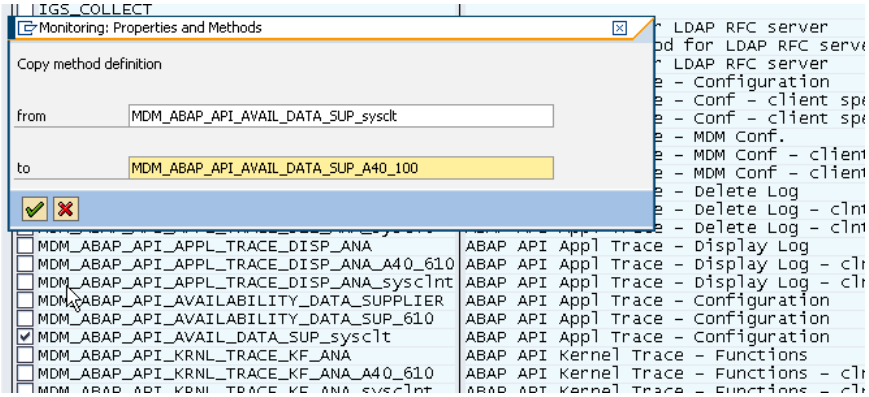
   Check the RFC definition as described in chapter 5.2.1.

   Mark *Method definitions*



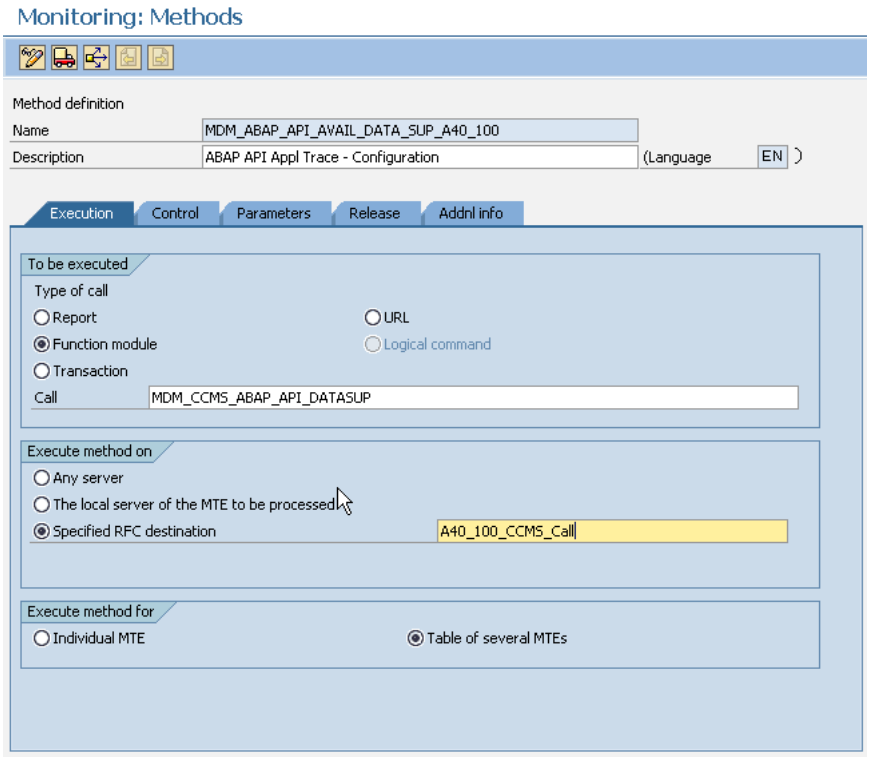
2. Select the delivered method “MDM_ABAP_API_AVAIL_DATA_SUP_sysclt” and copy it to a client-specific method (e.g. MSM_ABAP_API_AVAIL_DTA_SUP_A40_100).

   Choose “Change” to edit the new method.



3. On the Execution tab in the area “Execute method on”, select “Specified RFC destination” and enter a valid RFC destination to the specific client (here A40_100_CCMS_Call).

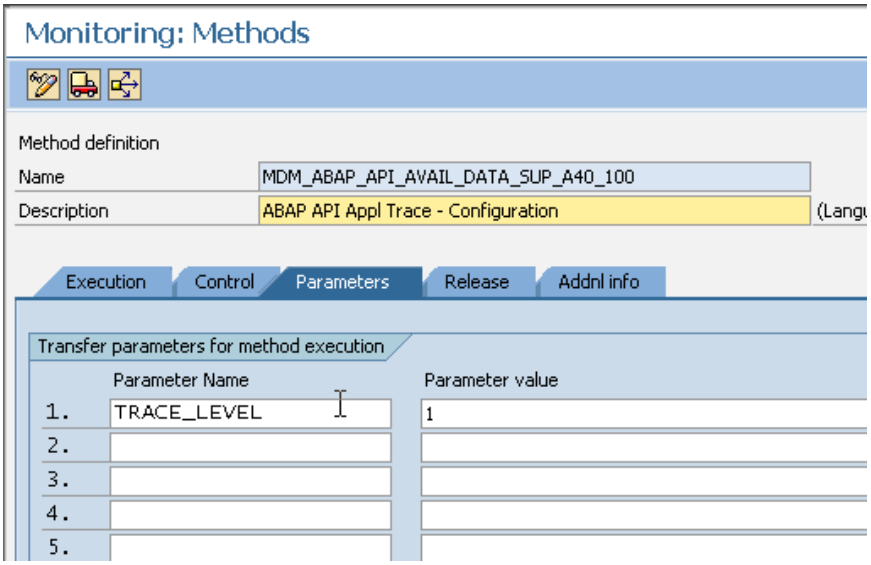
   Note:
   To define the RFC destination, see section 5.2.1 about MDM ABAP API logging and tracing methods.

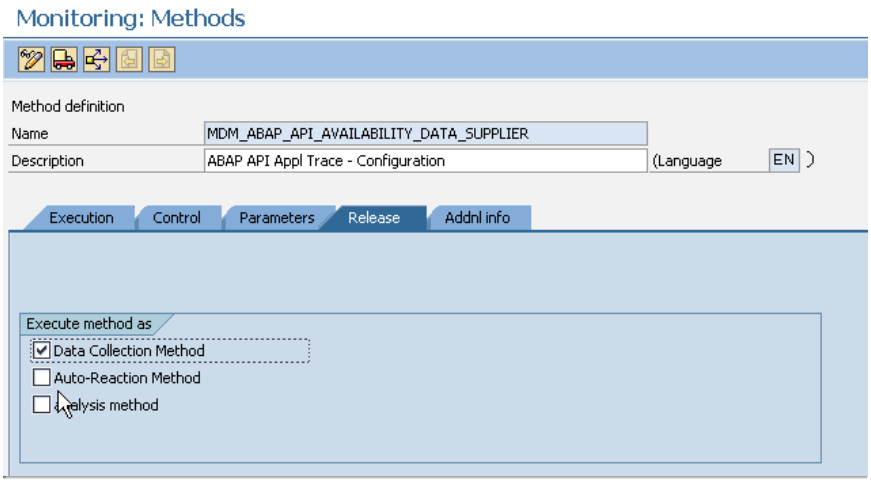


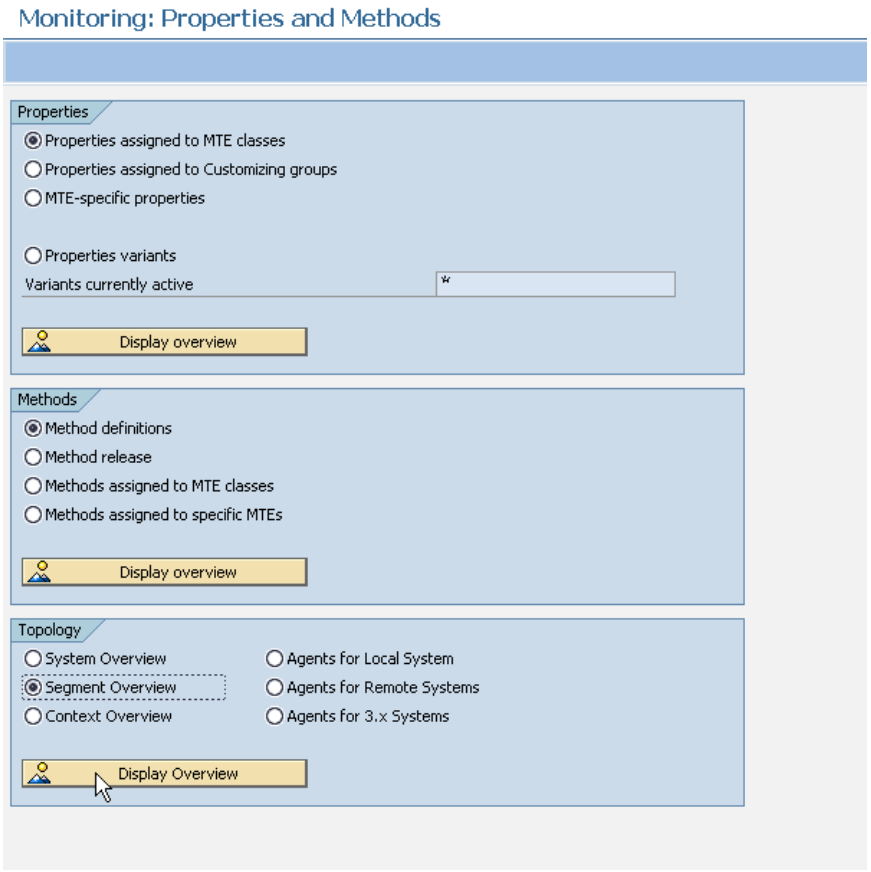
4. Go to the “Parameters” tab to define the parameters.

   Switch to Change Mode and enter the parameter definition. Save your entry.

   Check and if necessary reenter the method release definition on the Release tab in this transaction.







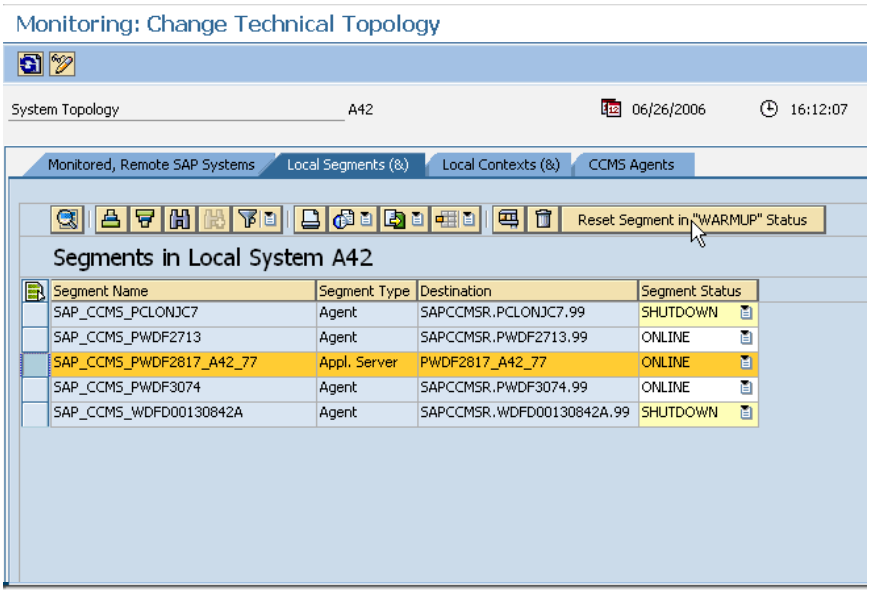
5. Monitored System:

   Start automatic method scheduling for the CCMS methods defined for “automatic start”.

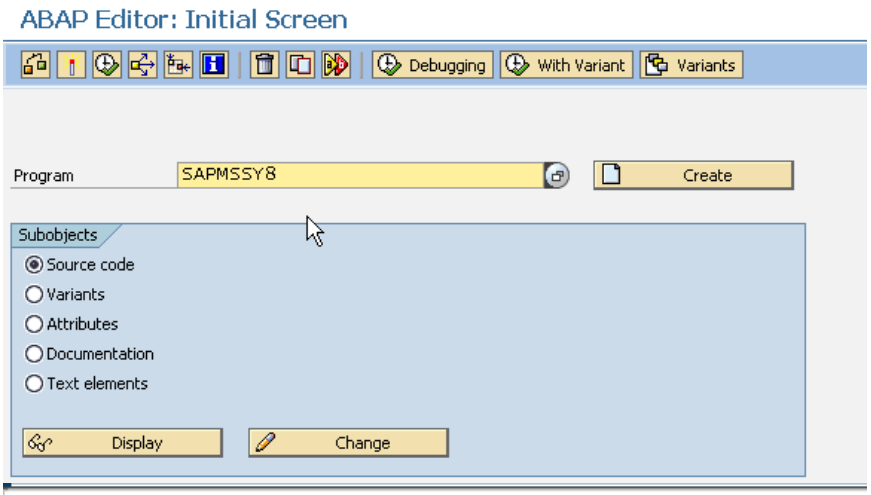
   In transaction RZ21, mark “Segment Overview” to restart the local CCMS segment of the monitored system.

   To do this, switch to Change mode, select the local application server, and choose “Reset Segment in “Warmup” Status”.

   If your data collector methods are correctly defined, the data suppliers are now executed.



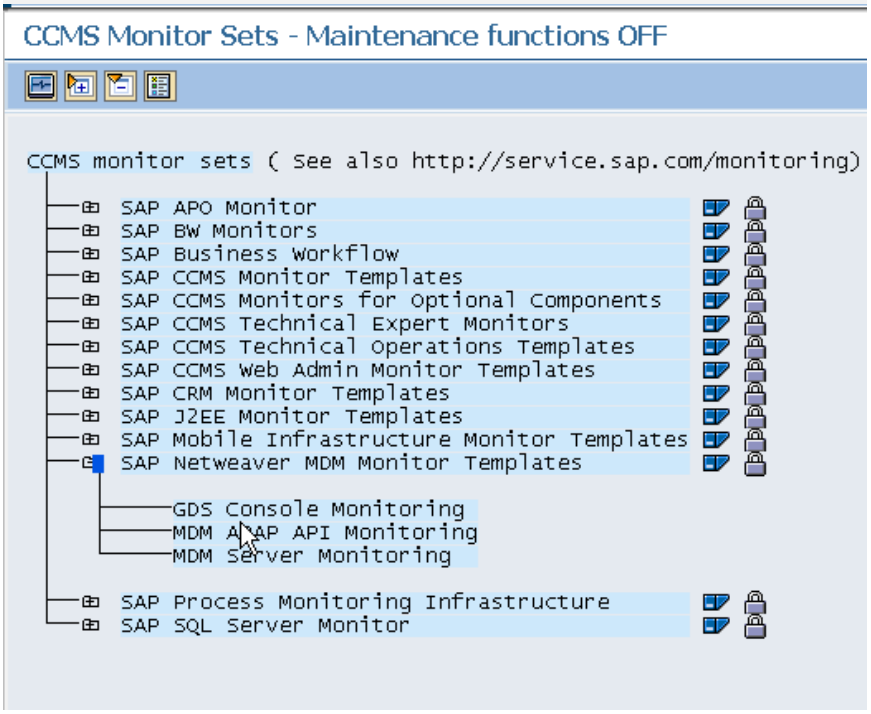
6. To make sure that all job scheduling is correct, go to transaction SE38 and start job SAPMSSY8 to schedule all regular CCMS jobs.



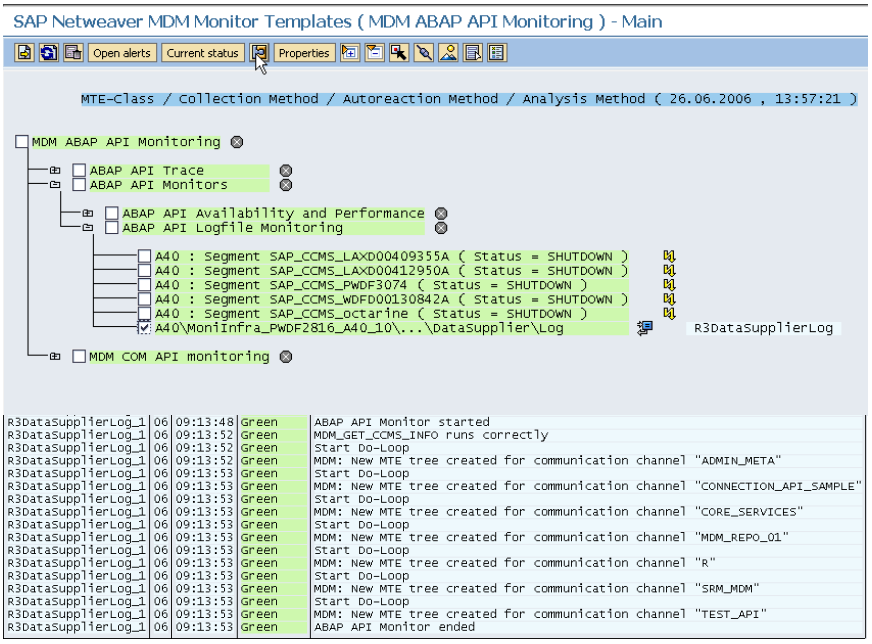
7. Check the self-monitoring protocol in transaction RZ20.

   Start MDM ABAP API Monitoring

   Go to “ABAP API Monitors”, “ABAP API Logfile Monitoring” and select the value ….\DataSupplier\Log for your monitored system. Call “Display Details” to display the self-monitoring log file for the CCMS.



8. Here you see an example of the MDM Availability Monitoring Data Supplier running with log trace level STANDARD.



R3DataSupplierLog_1	06	09:13:48	Green	ABAP API Monitor started
R3DataSupplierLog_1	06	09:13:52	Green	MDM_GET_CCMS_INFO runs correctly
R3DataSupplierLog_1	06	09:13:52	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:52	Green	MDM: New MTE tree created for communication channel "ADMIN_META"
R3DataSupplierLog_1	06	09:13:53	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:53	Green	MDM: New MTE tree created for communication channel "CONNECTION_API_SAMPLE"
R3DataSupplierLog_1	06	09:13:53	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:53	Green	MDM: New MTE tree created for communication channel "CORE_SERVICES"
R3DataSupplierLog_1	06	09:13:53	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:53	Green	MDM: New MTE tree created for communication channel "MDM_REPO_01"
R3DataSupplierLog_1	06	09:13:53	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:53	Green	MDM: New MTE tree created for communication channel "R"
R3DataSupplierLog_1	06	09:13:53	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:53	Green	MDM: New MTE tree created for communication channel "SRM_MDM"
R3DataSupplierLog_1	06	09:13:53	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:13:53	Green	MDM: New MTE tree created for communication channel "TEST_API"
R3DataSupplierLog_1	06	09:13:53	Green	ABAP API Monitor ended

9. Here you see the same Data Supplier running with trace level ENHANCED. As you see, each CCMS call is logged. This is useful, for example if the Data Supplier does not finish successfully.

   Example 2 shows level ENHANCED for the Advanced Tracing Data Supplier.

R3DataSupplierLog_1	06	09:15:25	Green	ABAP API Monitor started
R3DataSupplierLog_1	06	09:15:33	Green	MDM_GET_CCMS_INFO runs correctly
R3DataSupplierLog_1	06	09:15:33	Green	MDM: Monitor context MDM_ABAP_API_TOPNODE successfully created: User: BREITER
R3DataSupplierLog_1	06	09:15:33	Green	MDE Server Logon was succesful: At least one comm channel available
R3DataSupplierLog_1	06	09:15:33	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:15:33	Green	MDM: New MTE tree created for communication channel "ADMIN_META"
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_COM_CHAN_CL "#EC NOTEXT ADMIN_META" su
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_AVAIL_CL ADMIN_META" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_MO_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEARTBEAT_CLASS ADMIN_META" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SMES_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_STATUS_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_INTERV_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_AVAIL_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SMES_REPORT_T100_MESSAGE for MTE "MDM_ABAP_API_HEART_STATUS_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_PERF_CL ADMIN_META" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_MO_CREATE_ATTACH for MTE: "MDM_ABAP_API_PERF_SUB_CL ADMIN_META" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_LASTRESP_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_REPORT_VALUE for MTE "MDM_ABAP_API_LASTRESP_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:15:33	Green	MDM: New MTE tree created for communication channel "CONNECTION_API_SAMPLE"
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_COM_CHAN_CL "#EC NOTEXT CONNECTION_API
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_AVAIL_CL CONNECTION_API_SAMPLE" succes
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_MO_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEARTBEAT_CLASS CONNECTION_API_SAMPLE"
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SMES_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_STATUS_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_INTERV_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_AVAIL_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SMES_REPORT_T100_MESSAGE for MTE "MDM_ABAP_API_HEART_STATUS_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_PERF_CL CONNECTION_API_SAMPLE" success
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_MO_CREATE_ATTACH for MTE: "MDM_ABAP_API_PERF_SUB_CL CONNECTION_API_SAMPLE" succ
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_LASTRESP_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_REPORT_VALUE for MTE "MDM_ABAP_API_LASTRESP_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	Start Do-Loop
R3DataSupplierLog_1	06	09:15:33	Green	MDM: New MTE tree created for communication channel "CORE_SERVICES"
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_COM_CHAN_CL "#EC NOTEXT CORE_SERVICES"
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_AVAIL_CL CORE_SERVICES" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_MO_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEARTBEAT_CLASS CORE_SERVICES" successf
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SMES_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_STATUS_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_INTERV_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_HEART_AVAIL_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SMES_REPORT_T100_MESSAGE for MTE "MDM_ABAP_API_HEART_STATUS_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "MDM_ABAP_API_PERF_CL CORE_SERVICES" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_MO_CREATE_ATTACH for MTE: "MDM_ABAP_API_PERF_SUB_CL CORE_SERVICES" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_CREATE_ATTACH for MTE: "MDM_ABAP_API_LASTRESP_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	MDM: SALI_PERF_REPORT_VALUE for MTE "MDM_ABAP_API_LASTRESP_CLASS" successful
R3DataSupplierLog_1	06	09:15:33	Green	Start Do-Loop
R3DataSupplierLog_1	06/27/2006	10:06:20	Green	ABAP API Monitor (Tracing) started
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: Monitor context A40_100_MDM_ABAP_API_TRACE successfully created: User: BREITER
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "A40_100_ABAP API Application Trace " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "ABAP API Application Trace - Configurati " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "ABAP API Application Trace - Display " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "ABAP API Application Trace - Delete " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "ABAP API Appl Trace - Appl Log Config " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "A40_100_ABAP API Kernel Trace " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "ABAP API Kernel Trace - Work processes " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	MDM: SALI_SUM_CREATE_ATTACH for MTE: "ABAP API Kernel Trace - Kernel Functions " successful
R3DataSupplierLog_1	06/27/2006	10:06:31	Green	ABAP API Monitor (Tracing) successfully ended

10. Here you see an example of an incorrect call of the data supplier for the availability check. No communication channel to the MDM Server is defined in this client !!

pplierLog_1	26.06.2006	13:59:53	Green	RSDSLAN1 started
pplierLog_1	26.06.2006	13:59:53	Green	RSDSLAN1 successfully completed
pplierLog_1	26.06.2006	14:01:27	Green	ABAP API Monitor started
pplierLog_1	26.06.2006	14:01:41	Green	No repository active
pplierLog_1	26.06.2006	14:01:41	Green	MDM: Monitor context MDM_ABAP_API_TOPNODE succe
pplierLog_1	26.06.2006	14:01:41	Red	Error in Logon to MDE Server No Communication
pplierLog_1	26.06.2006	14:01:41	Green	ABAP API Monitor ended

6 Example – Connecting a New System to the CEN MDM Monitor

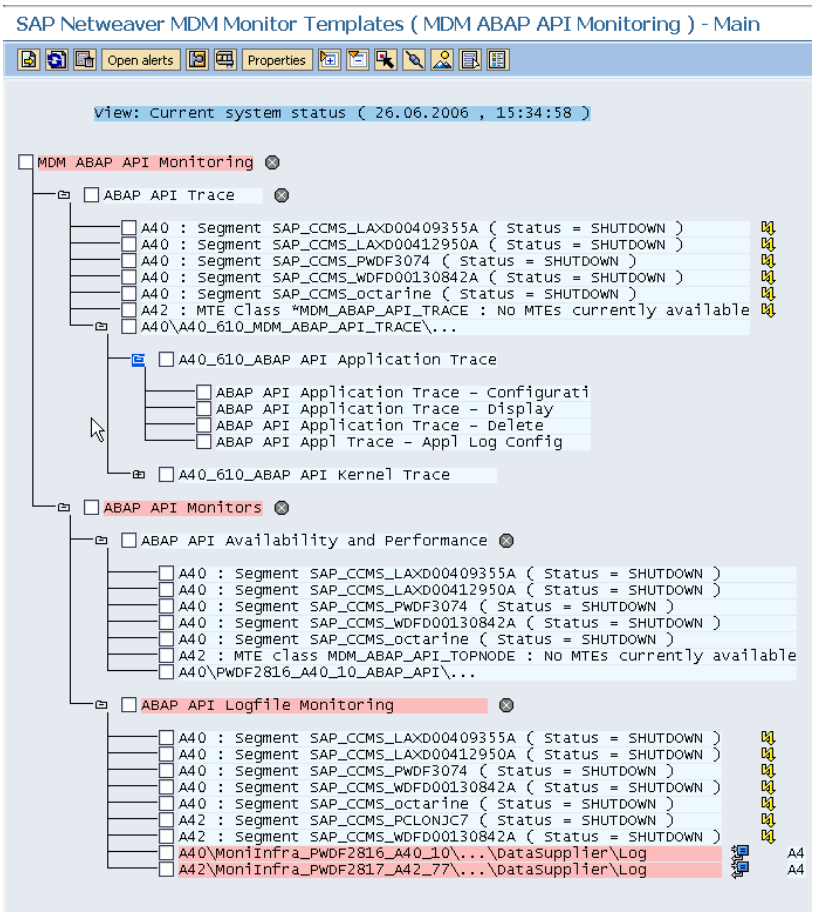
You followed all steps described in the previous chapters, but the new connected monitored system does not yet deliver any monitored data.

When you add a new monitored system, you might see the system in the MDM monitoring tree, but not see any MDM monitoring content. The reason is often that the MDM CCMS Data Suppliers (see chapter 5) have not yet been started.

In this case, the following activities might help to get the new system into the CEN:

1. In this screenshot you see that the monitored system A40(610) is defined and only the detailed tree for A40(610) is displayed.

   Add system A42 (002) as MDM ABAP API monitored system to the central monitoring system A42(610).

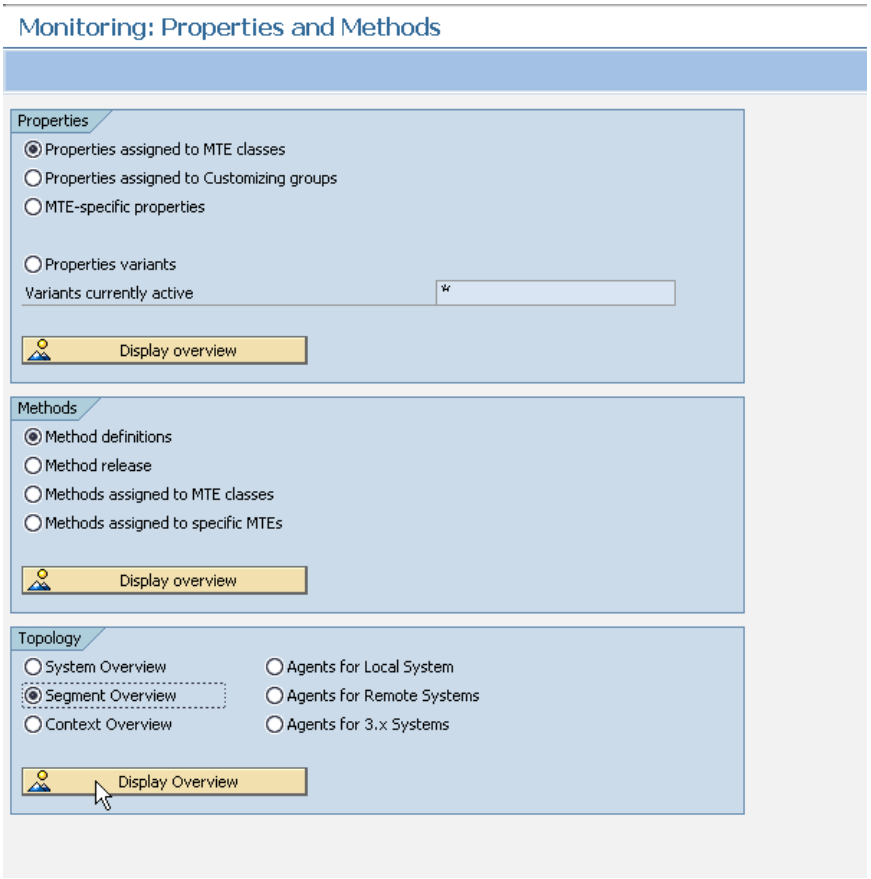


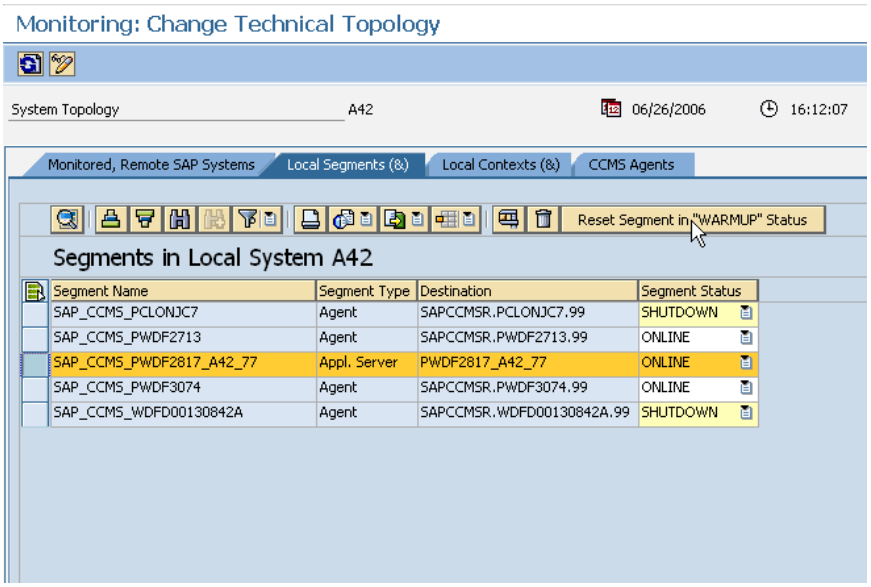
2. Log onto the monitored system, here A42(100).

   In transaction RZ21, mark *Segment Overview* to restart the local CCMS segment of the monitored system.

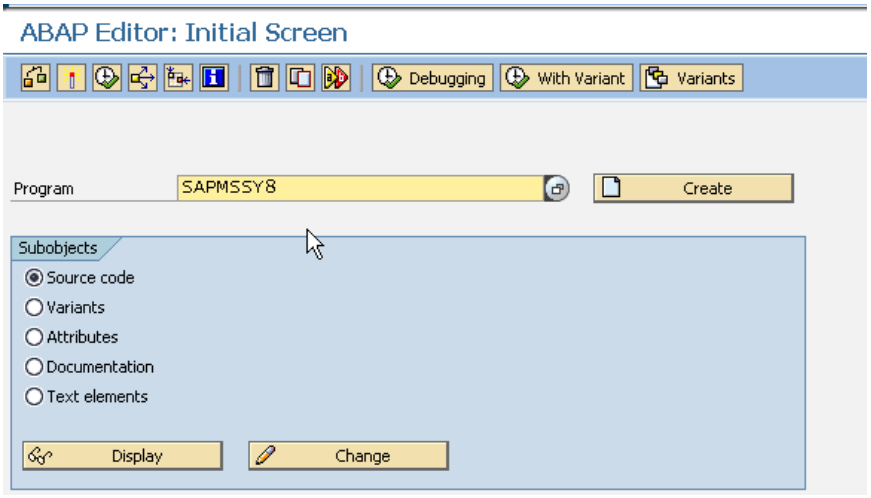
   Switch to Change mode, select the local application server and choose “Reset Segment in “Warmup” Status”.

   If your data collector methods are correctly defined, the data suppliers are now executed.



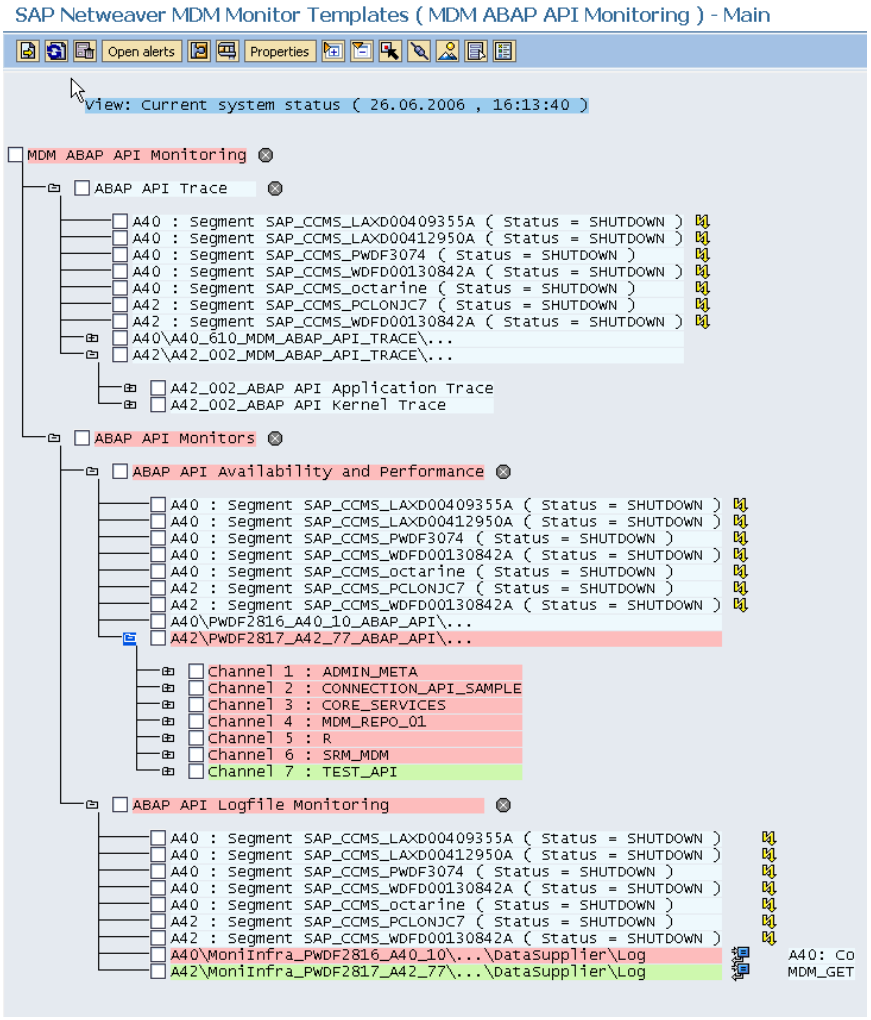


3. To make sure that all job scheduling is also correct, go to transaction SE38 and start program SAPMSSY8 to schedule all regular CCMS background jobs.



4. After CCMS communication has been set up, this data is transported to the central monitoring system (CEN: A42(610)).

   You see monitoring data for systems A40(610) and A42(002).



7 Monitored System (API) – Functionality and Usage

7.1 MDM ABAP API – Additional Documentation – Setup Guide and Advanced Tracing Usage

Setup and configuration have now been completed and the ABAP API Data Suppliers started.

Before the CCMS monitoring functionality can be checked, it is necessary to configure the ABAP API and set up communication channels between the ABAP system and an MDM Server.

For information about the setup and usage of the ***MDM ABAP API*** see the following How To Guides:

- **MDM ABAP API How To Guide Part 1 – Setting Up Customizing**
  This guide describes how to set up communication channels between the ABAP system and an MDM Server.
- **MDM ABAP API How To Guide Part 2 – Connect Repository**
  This guide describes how to connect to MDM repositories.

User Guide for "Advanced Logging and Tracing"

The following How To Guide describes how to use advanced logging and tracing for ABAP API software:

- **MDM ABAP API How To Guide CCMS – Advanced Logging and Tracing using CCMS**
  This guide also includes an ABAP API example program for checking the Advanced Tracing and Logging function.

For information about CCMS use of Advanced ABAP API Logging and Tracing, see the
User Guide for "MDM ABAP API Communication Channel Availability".

The next section of this guide describes how to use CCMS to monitor the availability of MDM ABAP API communication channels.

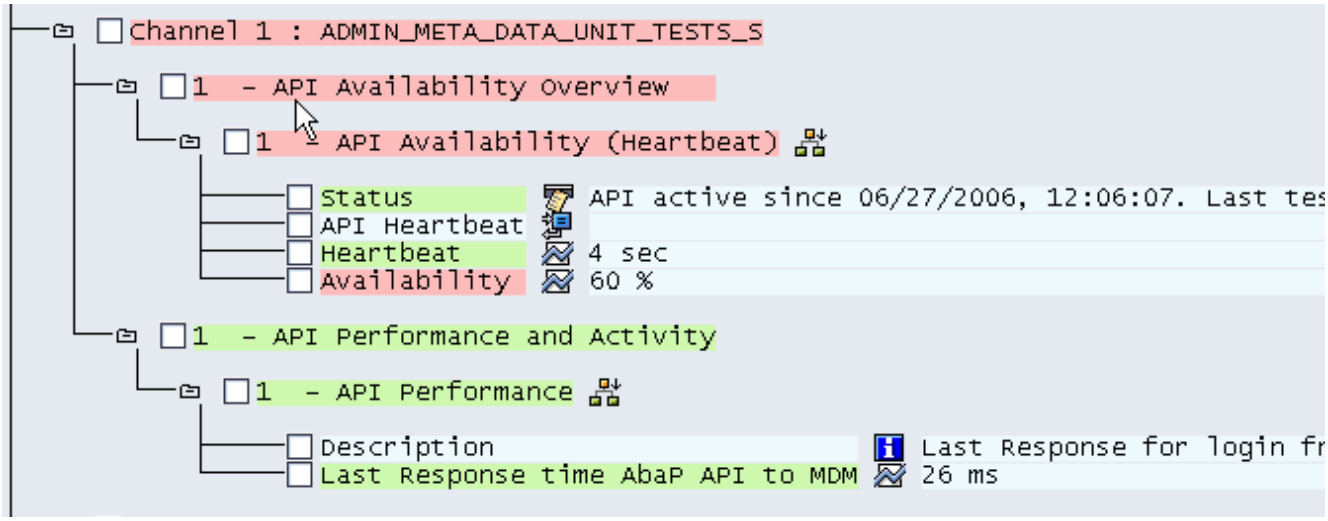
7.2 MDM ABAP API Availability Monitoring – User Guide

Prerequisite:

- You have configured the ABAP API communication channels and connected your ABAP system to an MDM Server.
- You have set up and configured CCMS monitoring (connected the CCMS agents and started the Data Suppliers).

You can find "Advanced Logging and Tracing" in the MDM CCMS monitor tree below the node "ABAP API Trace". "ABAP API Availability Monitoring" can be found in the parallel node "ABAP API Monitors".

Understanding Availability Monitoring in CCMS

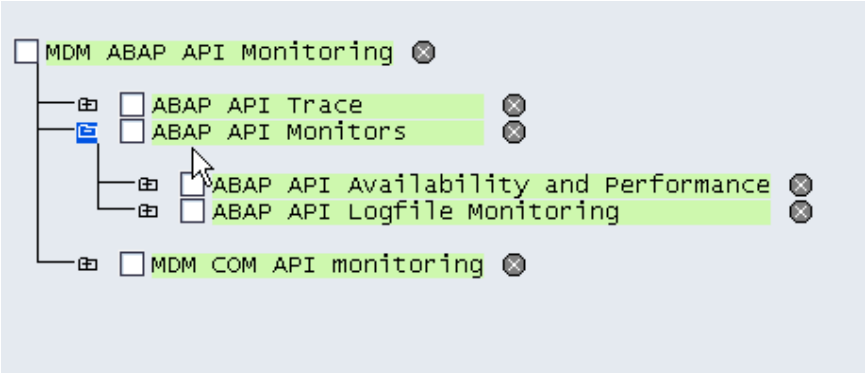


In MDM ABAP API customizing (transaction MDMAPIC), you defined a number of channels from the ABAP system to an MDM Server. Per channel you select the MDM Server and the MDM repository to be connected. This information is displayed in the ABAP API availability information for a specific client.

The following information is displayed per channel:

<no> - API Availability (Heartbeat)	No = <CCMS internal channel number>
Status	Defines if the channel is active or inactive since a given time. This value obviously fits the current availability value.
API Heartbeat	Delivered from CCMS, but not used
Heartbeat	Time since last availability check (in seconds)
Availability	Percentage of availability (per default: average for the last 15 minutes)
<no> - API Performance and Activity	No = <CCMS internal channel number>
<no> - API Performance	No = <CCMS internal channel number>
Description	
Last response time ABAP API to MDM	The response time that was measured when the MDM Server was called from ABAP for this channel.

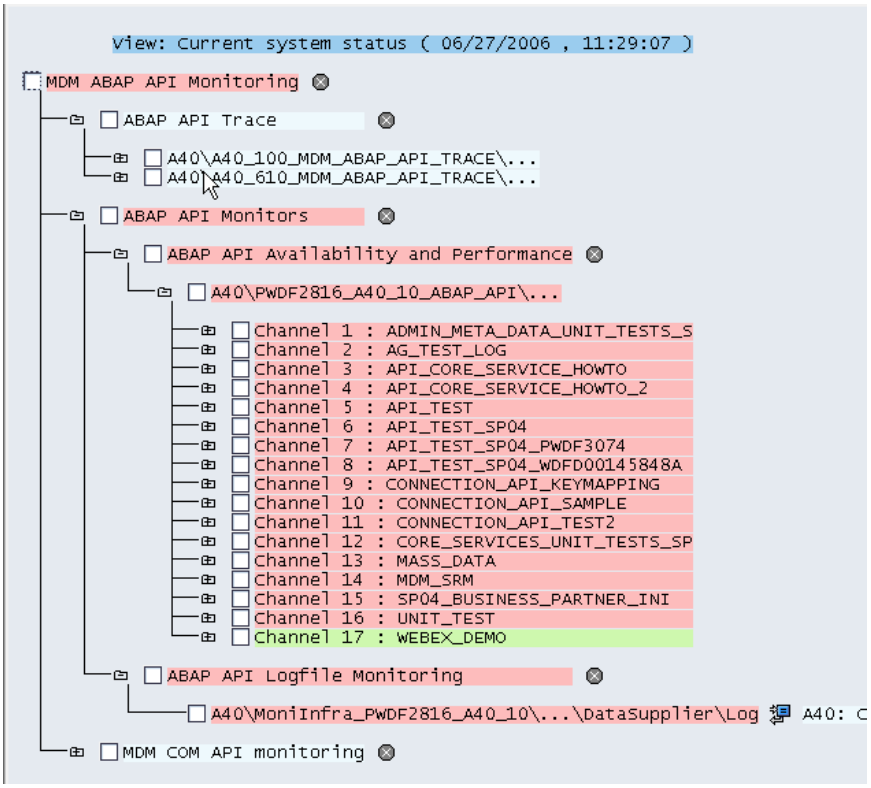
1. CEN: CCMS Transaction RZ20: This is the MDM Monitoring tree. All data for the Availability Monitor can be found in subtree ABAP API Monitors.



2. Example for a system in which 17 channels are defined in the ABAP API customizing.

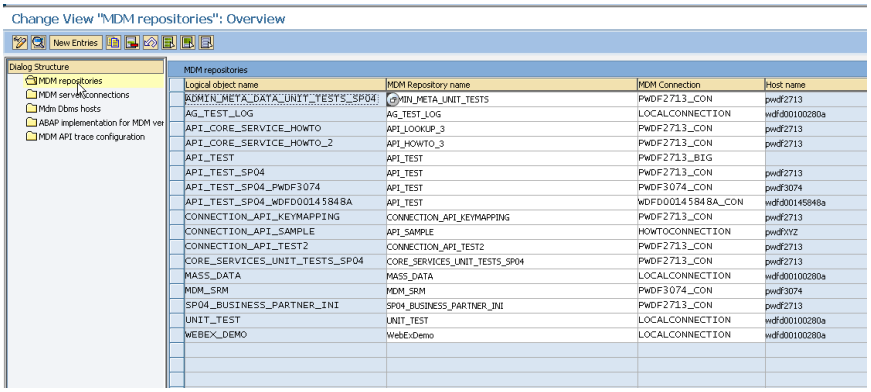
   The large number of channels displayed in red is due to the fact that the Data Supplier was just started, but the CCMS displays the availability average for the last 15 minutes.

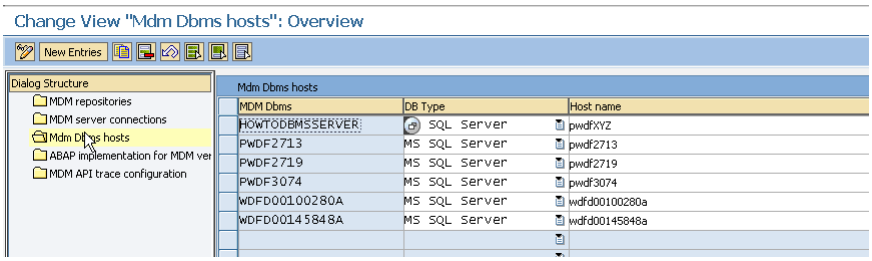
   One of the CCMS functions is to propagate the worst status from bottom to top, so if you see a tree node in red, you immediately know that something went wrong.



3. Enter your monitored system.

   You defined your communication channels in transaction MDMAPIC (see example screenshots).

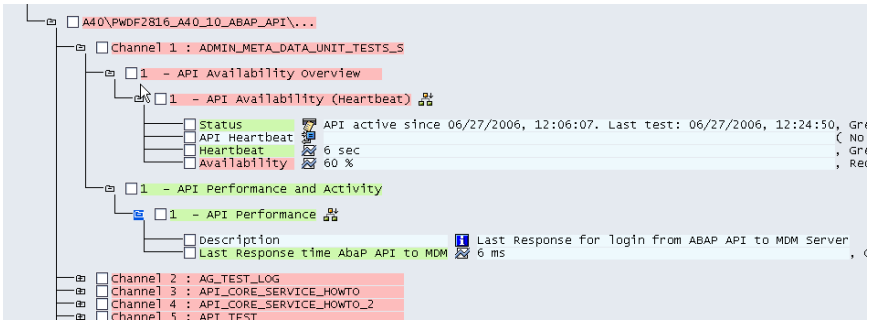




4. Change to the central monitoring system (CEN) to check the current status of the communication channels you defined.

   Start transaction RZ20 and enter the MDM Monitor tree.

   The screenshot shows the typical CCMS availability for one communication



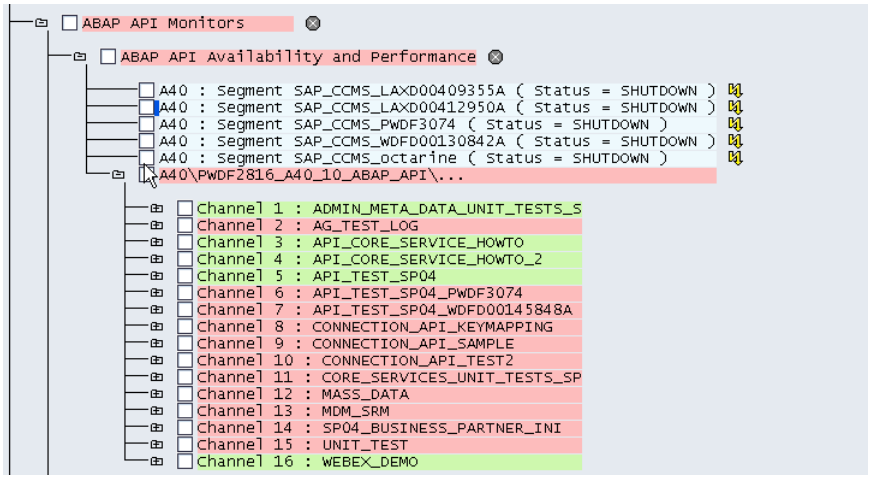
channel.

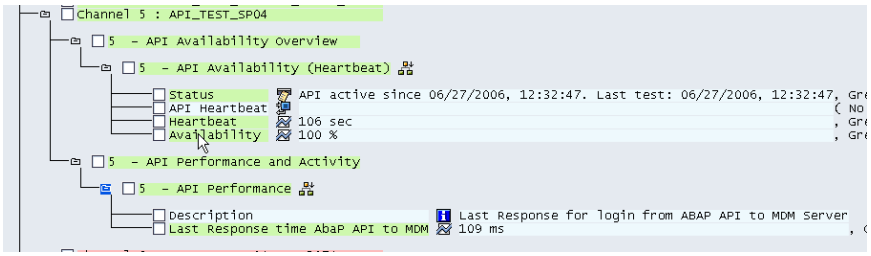
The repository on the MDM Server was started 8 minutes ago, so the average availability for the last 15 minutes is displayed as 60%. The thresholds are defined so that 60% availability sends an alert (red color).

You also see the real response time for the availability check (in this screen 6 ms)

5. After awhile the display changes and some channels show their availability, while others are not available.

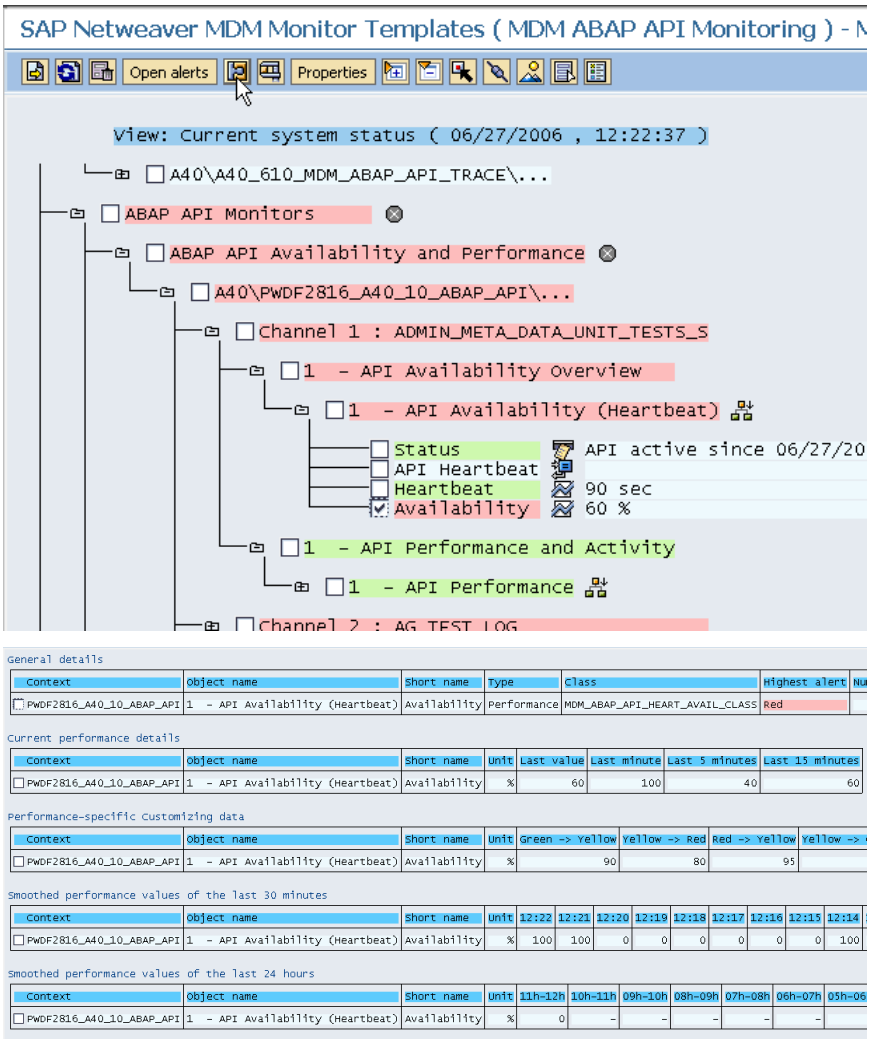
   You can enter an auto-reaction method for the Availability node you want to monitor and attach a method that sends an e-mail on alert.





6. Threshold settings for availability are defined in the Data Supplier, but can be overwritten in the CCMS monitor (see next chapter for threshold modifications).

   As you see in the detail data for the Availability monitoring tree element (MTE), the communication channel was available for the last 2 minutes but not before.



7.3 MDM ABAP API Availability Monitoring – Threshold Settings for Performance Attributes

Each monitoring tree element in the CCMS tree, that is a performance attribute such as “Availability”, has specific threshold settings that define the change from GREEN to YELLOW, YELLOW to RED and RED to YELLOW or YELLOW to GREEN status. RED status is directly linked to the alert mechanism of CCMS. If the status changes from YELLOW to RED, an auto-reaction method is therefore fired. You must define this behavior in the properties of the performance attribute.

You can adapt the delivered thresholds to the customer’s needs in two different places (see the CCMS documentation for details):

- **Data Supplier**:
  All performance attributes displayed in the MDM ABAP API availability monitoring tree are defined in the data supplier that is running on the ABAP API system
- **MTE properties**
  The tree can also be modified directly as shown in the following screenshots.

1. **Definition of Data Supplier:**

   The delivered data supplier for availability monitoring has a section for defining constants in which the thresholds are collected.

   The data supplier is a function called MDM_CCMS_ABAP_API_DATASUP.

   Start transaction SE37 to view the data supplier.

   Thresholds are pre-defined for three performance attributes:
   - Heartbeat Interval
   - Channel availability
   - Repository response time

```
constants:

* Dear Customer: Please Adapt the threshold data of all performance
*               data suppliers to your needs.

* Threshold constants for the Hearbeat Interval in seconds
  Con_Hearbeat_Intvl_Green2Yell          type i value 360,
  Con_Hearbeat_Intvl_Yell2Red            type i value 720,
  Con_Hearbeat_Intvl_Yell2Green          type i value 360,
  Con_Hearbeat_Intvl_Red2Yell            type i value 720,

* Threshold constants for the Availability over last 15 min in %
  Con_Availability_Green2Yell            type i value 90,
  Con_Availability_Yell2Red              type i value 80,
  Con_Availability_Yell2Green            type i value 85,
  Con_Availability_Red2Yell              type i value 95,

* Threshold constants for the MDM Repository response time in ms
  Con_MDMRep_RespTime_Green2Yell         type i value 450,
  Con_MDMRep_RespTime_Yell2Red           type i value 600,
  Con_MDMRep_RespTime_Yell2Green         type i value 400,
  Con_MDMRep_RespTime_Red2Yell           type i value 550.
```

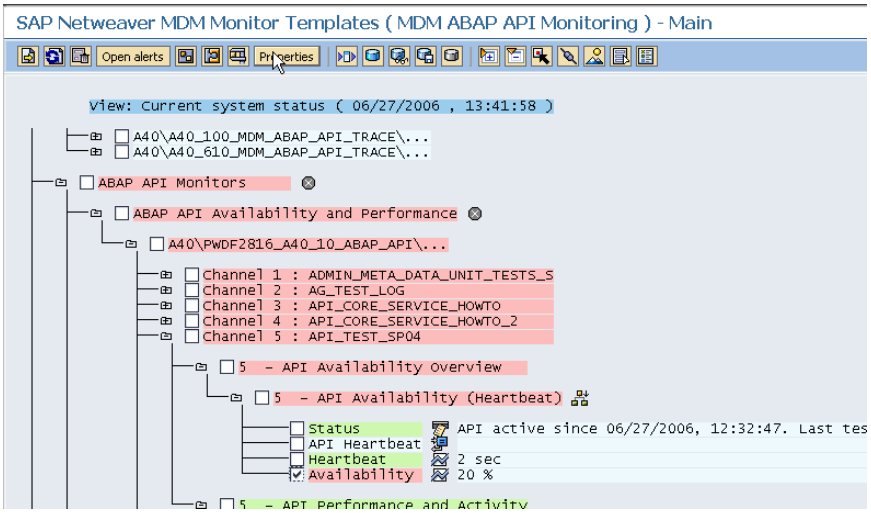
2. **Definition of MTE properties:**

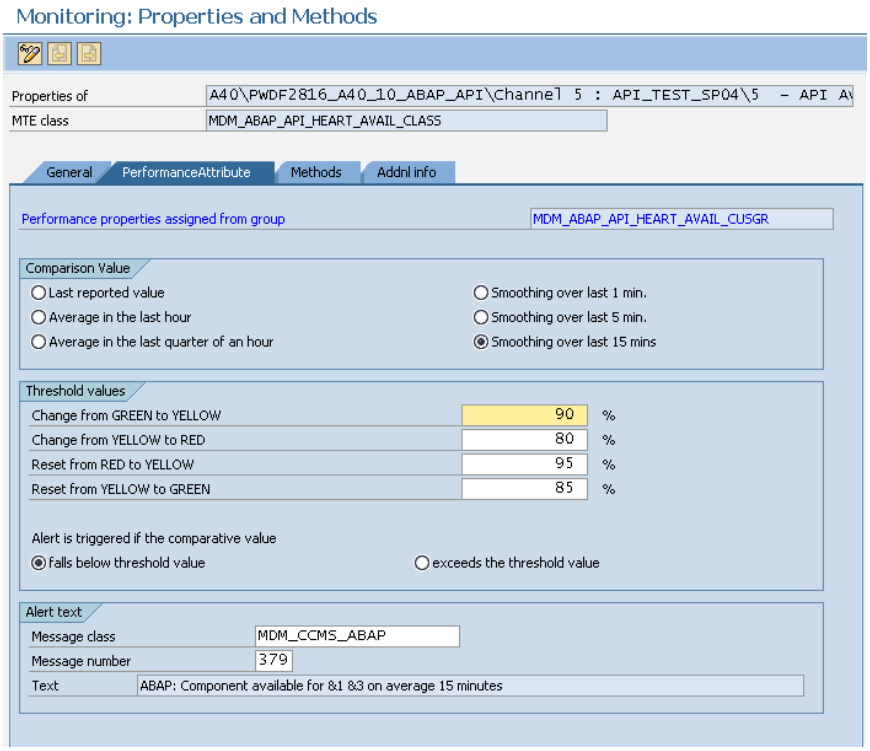
   You want to change the delivered thresholds for the Availability parameter. Select the Availability tree node and choose Properties.

   Switch to Change mode, modify the properties, and choose tab “PerformanceAttribute”.

   Here you can change the thresholds values as well as the comparison value for the evaluation. “Smoothing over last 15 minutes” is set by default.

   You can also change the Alert text. Use a message stored in an ABAP message class.





7.4 Additional CCMS Functions

The CCMS delivers a number of additional functions that allow you to use the CEN as control center. Two very important functions will be discussed here. For greater detail, see the standard documentation for CCMS.

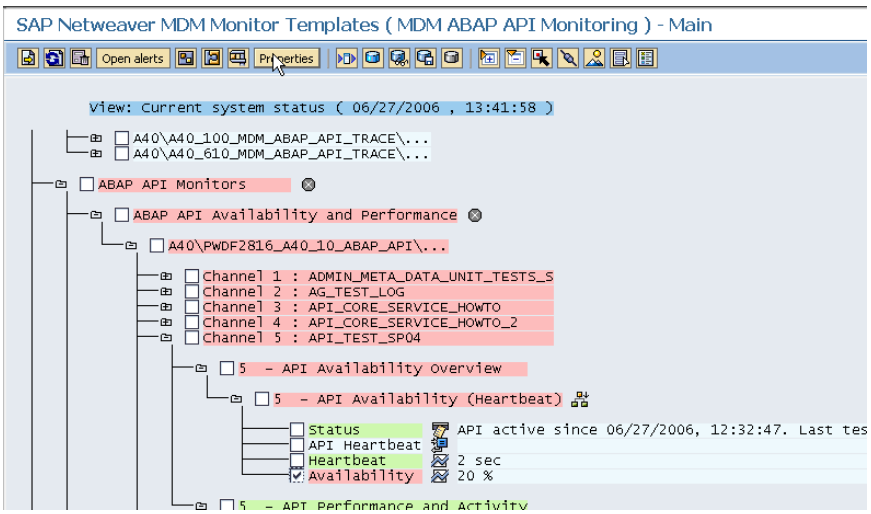
7.4.1 Sending an E-Mail on Alert

This function is used very frequently and is one of the most important CCMS capabilities.
CCMS allows you to define an “auto-reaction” method for your monitoring element. One possible auto-reaction is CCMS_SEND_EMAIL_ON_ALERT.
The following CCMS documents provide details about auto-reaction as well as a list of SAP notes:

Note	176492:	Automatic e-mail when an alert occurs (RZ20)
	429265:	CCMS monitor architecture: Central Auto-Reaction
	502959:	RZ20: External e-mails as auto-reaction
	617547:	RZ20: Sending alerts as mail and SMS
	…	

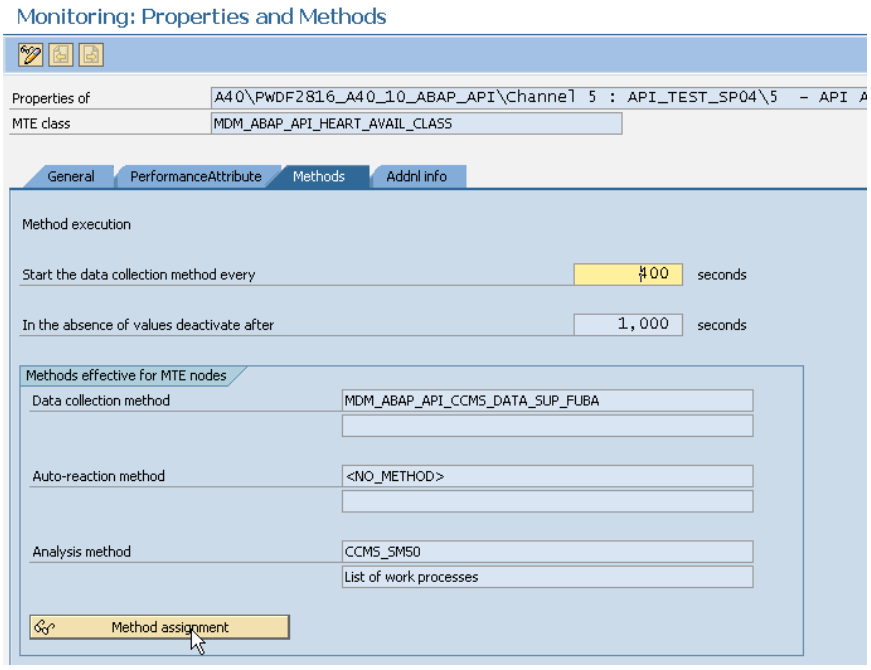
You can define the auto-reaction in the Properties screen for the monitoring element. You can also define the auto-reaction method in a Data Supplier.

1. Start RZ20 and select the monitoring tree element for which you want to send alert e-mails. Select the parameter and choose Properties.



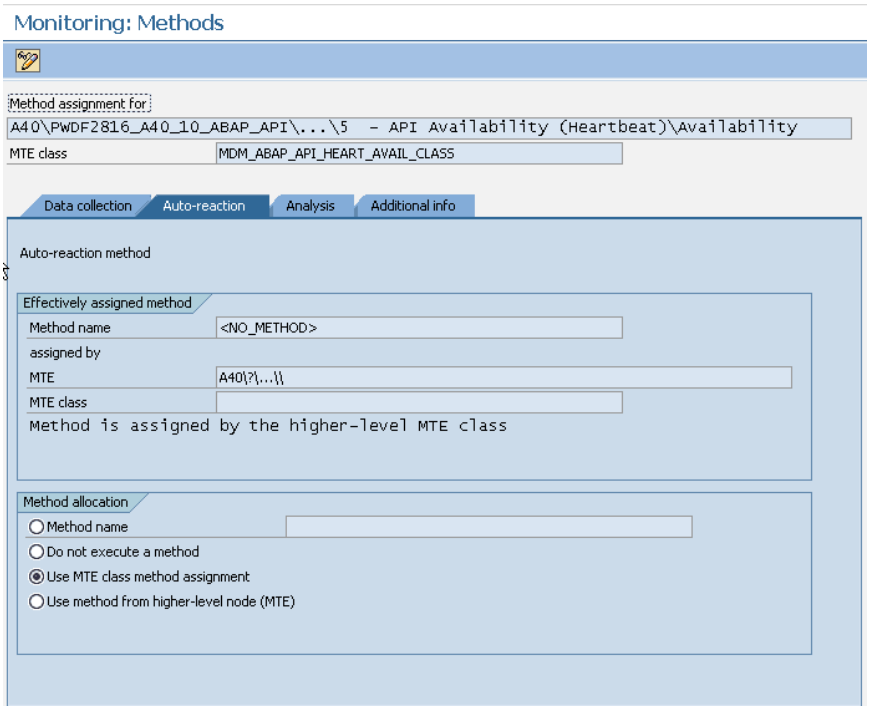
2. Choose the Methods tab to change the method assignment.

   Click on “Method Assignment” to enter the modification screens for attached methods.



3. Choose the *Auto-Reaction* tab and enter the previously defined e-mail alerting method (in transaction RZ21 – don't forget to release the method for auto-reaction).

   Save the method assignment.



7.4.2 Storing Performance Data in a Central History → Central Performance History (CPH)

The Central Performance History is an application connected to the CCMS in CEN. It has two main functions that are not provided in the CCMS.

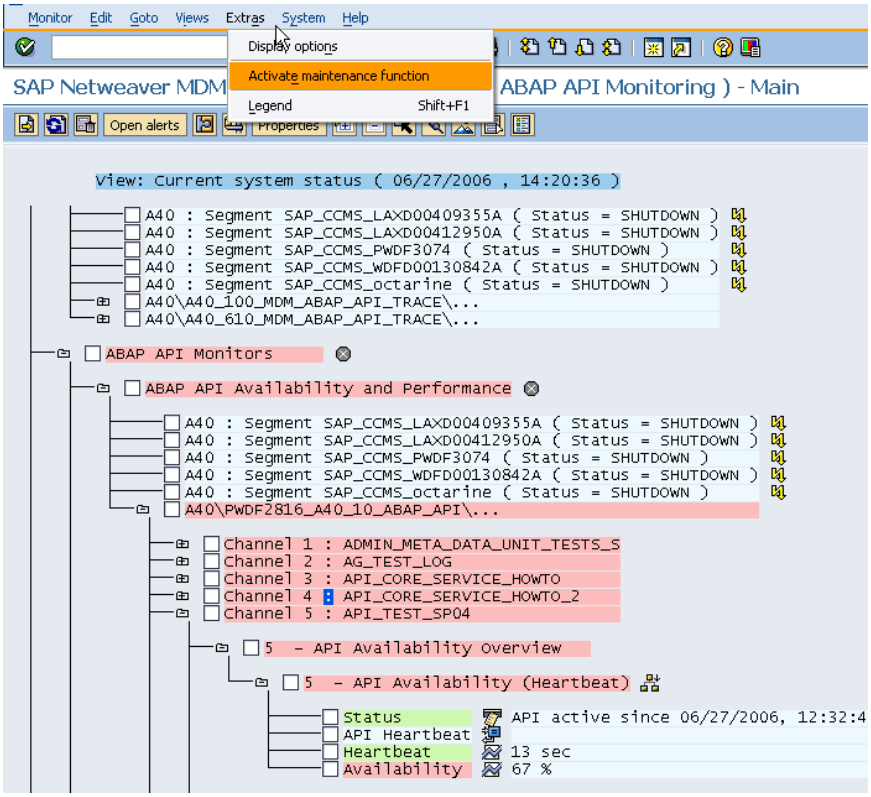
1. The CCMS stores performance data over a restricted time period, but not for longer time periods, due to the huge amount of data that grows over time. Specific performance criteria (defined by the monitoring user) can be collected over time, aggregated to other time units, and displayed using the SAP graphics application.

2. The MDM Server monitor delivers performance and availability data to the CEN, but does not calculate values such as min/max and average for the data. This can also be done with the CPH.

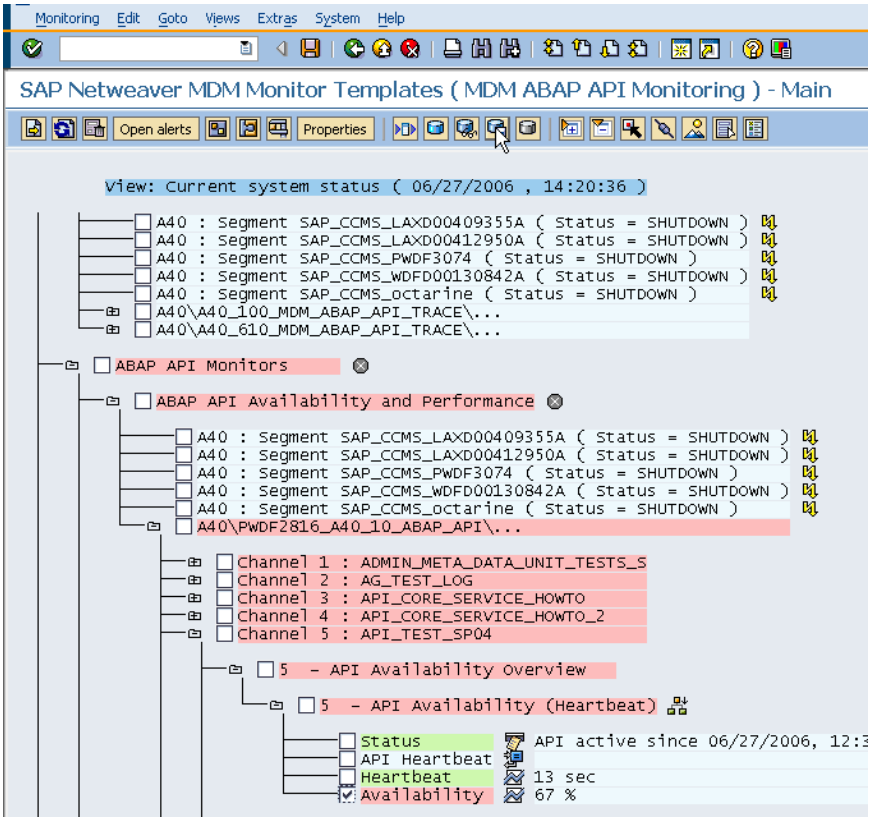
This guide gives a short introduction on how the CPH can be used for MDM performance and availability data. For details see the CCMS documentation.

1. In the MDM monitor tree, select the performance attribute you want to store in the CPH.

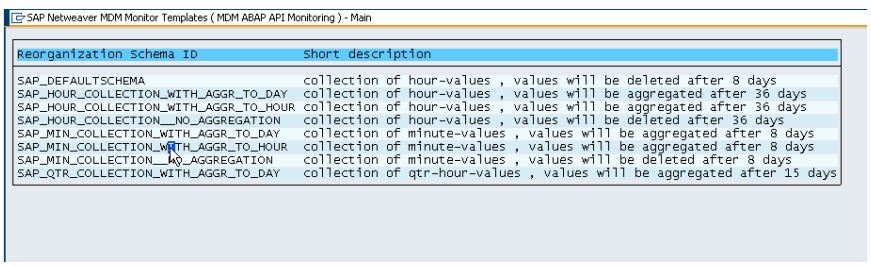
   Activate the CCMS maintenance functions.



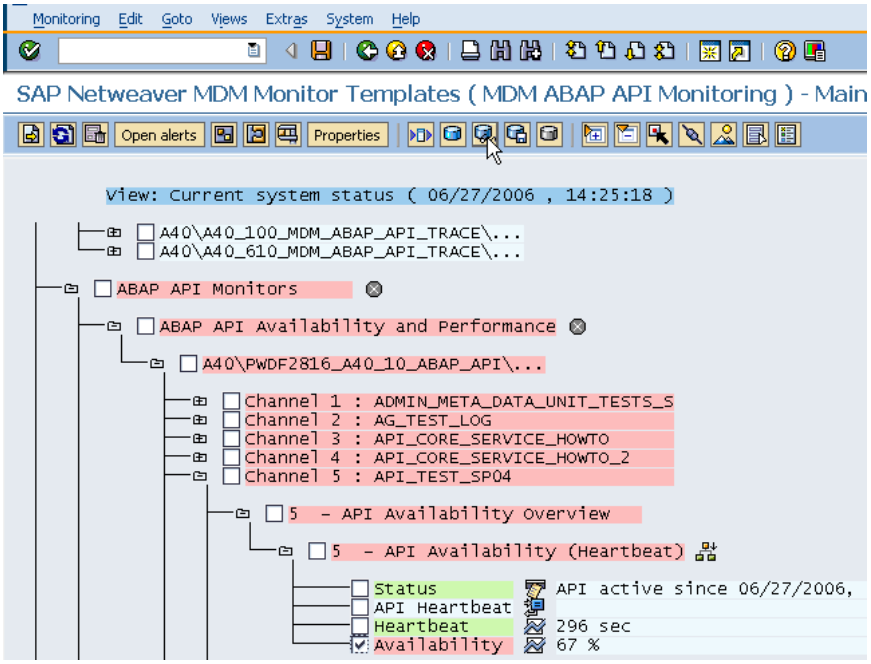
2. Select the performance attribute and choose “Collect MTE class in central performance history”.
   This definition is now valid for the MTE class and can be used more than once in the MDM tree.



3. Select the aggregation mode for your data, e.g. store minute-values but aggregate them to hours after a specific time interval.



4. To view the CPH history for your selected attribute, select the tree element again and choose “Display history of selected MTEs”.



For details see the CCMS documentation for CPH.

8 Appendix

8.1 Standard Monitoring Documentation

SAP Service Marketplace contains the complete set of standard documentation for CCMS monitoring: **`service.sap.com/Monitoring`** → *Monitoring in Detail.* The following guides are available:

SAP Service Marketplace contains the complete set of standard documentation for CCMS monitoring: **`service.sap.com/Monitoring`** → *Monitoring in Detail.* The following guides are available:

- Availability Monitoring and Agent CCMSPING
- CCMS Agents: Features, Installation and Usage
- CCMS System Component Repository
- Central Performance History of the Monitoring Architecture
- Customizing and Operating GRMG Monitoring SAP NW 04
- Customizing and Operating GRMG Monitoring SAP Web AS 6.20
- Design und Integration von SNMP-Funktionen in SAP NetWeaver
- Forwarding Alerts to Alert Management (ALM)
- Functional Trace (Transaction STATTRACE)
- Global Workload Monitor (Transaction ST03G)
- Integration of CPH Data into the Business Warehouse
- Java Monitoring API - Properties and Installation
- Configuration of the Monitoring Architecture
- Creating and Editing Monitors and Sets of Monitors
- Monitoring Jobs with the Alert Monitor
- Monitoring Multiple Systems
- Monitoring Response Times of Transactions or Clients
- Monitoring qRFC and tRFC Calls
- Preconfigured Monitors
- Predefined Auto-Reaction Methods of the Alert Monitor
- SAPOSCOL: Properties, Installation, and Operation
- Sending Alerts as SNMP Traps
- Technical Views of the Alert Monitor
- The SAP Expert Monitor for EMC (SEME)
- Windows Event Log Monitoring with CCMS Agents
- Workload Monitor (Transaction ST03N)

8.2 Important SAP Notes for Monitoring

SAP Note	Description
889366	*MDM55 SP03 ITSAM: LogMon Monitoring – Examples*
889580	*MDM55 SP03 ITSAM: CCMS Monitoring: MDM template*
889579	*MDM55 SP03 ITSAM: Procmon Monitoring MDM Server – Example*
436186	*Installing saposcol as a service*
618053	*Download Location for NTSCMGR.EXE*
548699	*FAQ: OS Collector SAPOSCOL*
957036	*MDM55 SP04 ITSAM: Logmon Monitoring – Examples*
956783	*MDM55 SP04 ITSAM: CCMS Monitoring: MDM Template*
1272117	*MDM 7.1 ITSAM: Procmon Monitoring MDM*