Enabling GRMG Heartbeat Monitoring

25. März 2003 – Version 1.0

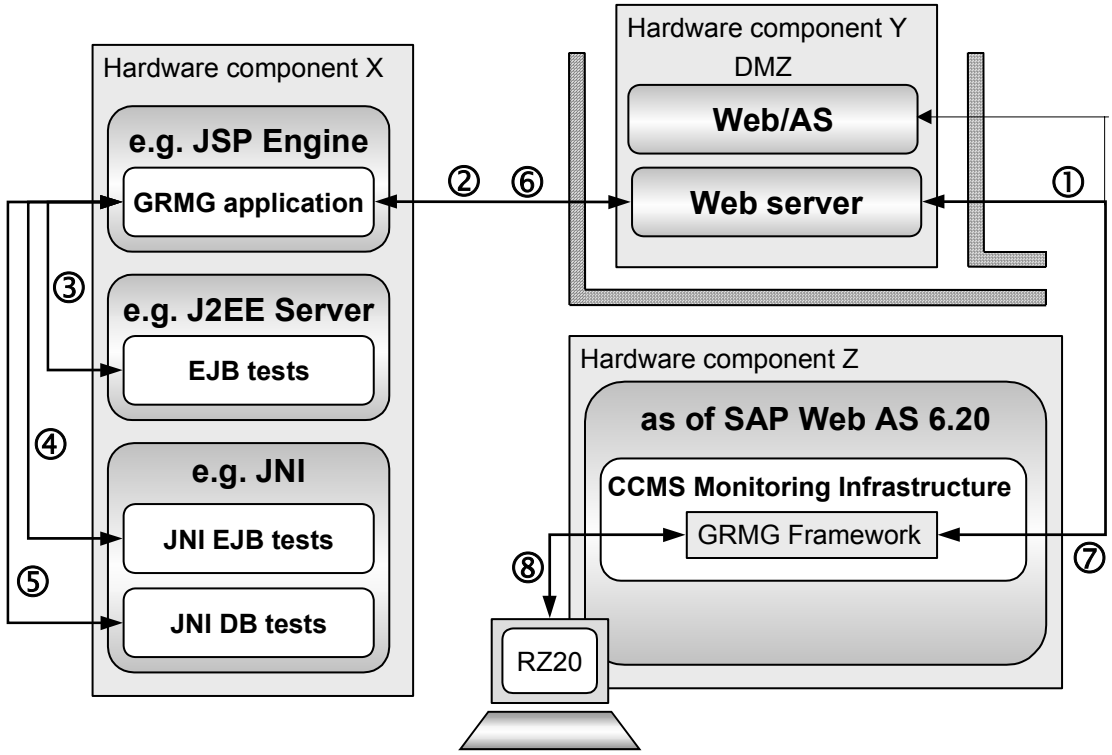
Table of content:

1 Project goal

The goal of the Generic Request and Message Generator (GRMG) is to test the availability of a chain of technical components, of a chain of business steps, or of a mixture of both. In this way, it can be used for technical monitoring as well as for application monitoring. Basically, technical monitoring and application monitoring are necessary to ensure the availability of business processes / business scenarios.

The GRMG consists of two parts, the GRMG Framework and the GRMG Application. Both parts are absolutely necessary to run a GRMG environment. The GRMG Framework is embedded in the Computing Center Management System (CCMS) Monitoring Infrastructure and maintained by SAP's CCMS development group / SAP Active Global Support. The task of the GRMG Framework is to send a test request (GRMG Request) against a test scenario (GRMG Application), to receive the response (GRMG Response) and to interpret the response as a CCMS monitoring tree. The GRMG Application is part of the tested application and is maintained by SAP's development departments. The GRMG Application is a callable application (JSP, Servlet, or standard SAP ABAP Objects class) with a well-defined interface (GRMG Request). The GRMG request and GRMG response are messages in a well-defined XML based format.

GRMG Test Scenario – Concept

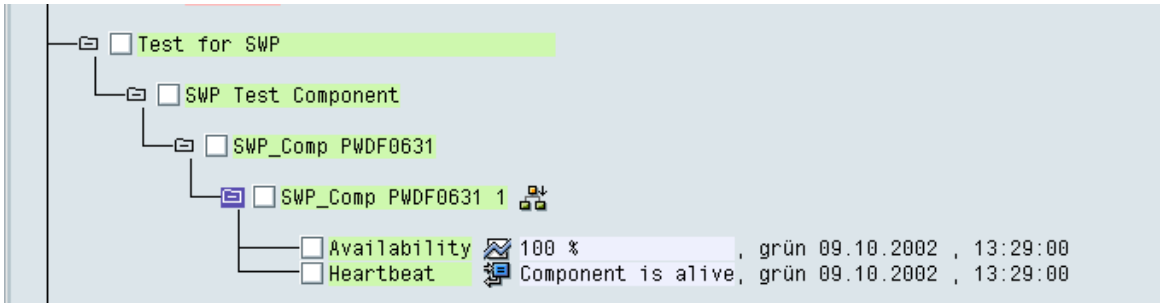


The figure above provides an example of a GRMG test scenario. The GRMG Framework periodically calls a GRMG Application that tests the availability of a chain of different software components (such as J2EE Servers or Web/Application Servers, as of SAP_BASIS Release 6.20) or technical functions (such as JNI). The GRMG Application is implemented as a JAVA Servlet in the case of a J2EE server or as a standard SAP ABAP Objects class in the case of a Web/AS.

In a typical web server scenario, the Web Server (1) transfers the GRMG request, which is transmitted by the HTTP POST method, and is then processed by the Servlet Engine (2). The tests for the different technical components / business process steps are implemented within the GRMG Application. The GRMG Application checks the availability of EJB (3), of the JNI EJB (4) and of the JNI DB (5). The results of the different checks are collected by the GRMG Application and combined in a GRMG Response. The GRMG Reponse is transferred back to the GRMG Framework (6) and is accordingly interpreted by the GRMG Framework into a CCMS monitoring tree.

It is important to note that the GRMG application is best implemented as an agent in a web server scenario. GRMG is designed with the agent concept - a level of indirection between the GRMG infrastructure and the tested components - in mind. This design allows GRMG to distinguish in its responses between failures of the agent (failures of the scenario itself, such as communication failures or agent down) and failures of the components that are tested by the scenario. A communication failure does not necessarily mean that the tested components are unavailable to the users who need them. However, a GRMG application may also be part of a tested application or technical component itself.

In a Web/AS scenario, the GRMG request is POSTed to a service in an SAP Web/AS. The standard handler class for the service allows the scenario to specify that certain function modules be carried out to test technical or application-level functionality in the receiving system or in other SAP Systems or components. The ABAP class then returns the results of any tests in a GRMG Response. As in web server scenarios, the GRMG infrastructure interprets the Response and present it as a monitoring tree in the CCMS Monitoring Architecture. The ABAP class allows you to use one technology, GRMG, to monitor all of the components in an application, whether they are ABAP or Java based.



The figure above provides an example of a CCMS monitoring tree. The structure of the CCMS monitoring tree is Component, Component / Host, Component / Host / Instance, and availability in percent and message log. You may create and run as many GRMG scenarios as you wish, each of which specifies a single URL or RFC destination of type HTTP. Each GRMG scenario may report on as many components as you wish to test from the GRMG application triggered by the scenario.

Using CCMS alerting you can see whether a technical component / business process step is available or not. The availability monitoring attribute reports on the percentage of time that the monitored component signaled a condition of 'OKAY' – component up and ready for use. The availability metric can be captured in the CCMS Performance Database for long-term reporting or service-level (SLA) verification. The message log lets a GRMG application return one or more messages per GRMG scenario execution (only one message in the case of the ABAP GRMG application). Each message can be separately rated and can trigger a separate CCMS alert, a trouble-ticket for the system or application administrator.

2 GRMG Framework

The parts of GRMG Framework are GRMG Customizing, GRMG Request Processing, and GRMG Response Processing. The different parts are described within the next section.

2.1 GRMG Customizing

For administration purposes and to transfer properties from the GRMG Framework to the GRMG Application it is necessary to maintain GRMG-specific Customizing. The GRMG Customizing consists of the tables GRMG_CONTROL, GRMG_SCENARIOS, GRMG_SCENARIOS_T, GRMG_COMPONENTS, GRMG_COMPONENT_T, and GRMG_PROPERTIES. Within the GRMG Framework there is no specific Customizing transaction. Instead, you maintain customizing in a file which you then upload into the SAP System that will run the scenarios.. The Customizing file is in a well-formatted XML-based format on your desktop and can be uploaded using transaction GRMG => Upload GRMG Customizing. Small corrections can be made to uploaded customizing in transaction SE16, directly in the tables. However, direct editing may

also result in capitalization of URLs, which may in turn produce scenario failures if paths to scripts or query-string data are case-sensitive.

An example XML-based Customizing file and the appropriate DTD are shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<customizing>
          <control>
               <grmgruns> .. </grmgruns>
               <runlog> .. </runlog>
               <errorlog> .. </errorlog>
          </control>
          <scenarios>
               <scenario>
                    <scenname> .. </scenname>
                    <scenversion> .. </scenversion>
                    <sceninst> .. </sceninst>
                    <scentype> .. </scentype>
                    <scenstarturl> .. </scenstarturl>
                    <scenstartmod> .. </scenstartmod>
                    <scentexts>
                         <scentext>
                              <scenlangu> .. </scenlangu>
                              <scendesc> .. </scendesc>
                         </scentext>
                         …
                    <scentexts>
                    <components>
                         <component>
                              <compname> .. </compname>
                              <compversion> .. </compversion>
                              <comptype> .. </comptype>
                              <comptexts>
                                   <comptext>
                                        <complangu> .. </complangu>
                                        <compdesc> .. </compdesc>
                                   </comptext>
                                   …
                              <comptexts>
                              <properties>
                                   <property>
                                        <propname> .. </propname>
                                        <proptype> .. </proptype>
                                        <propvalue> .. </propvalue>
                                   </property>
                                   …
                              </properties>
                         </component>
                         …
                    <components>
               </scenario>
               …
          </scenarios>
</customizing>

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT customizing (control, scenarios)>
<!ELEMENT control (grmgruns, runlog, errorlog)>
<!ELEMENT grmgruns (#PCDATA)>
<!ELEMENT runlog (#PCDATA)>
<!ELEMENT errorlog (#PCDATA)>
<!ELEMENT scenarios (scenario+)>
```

```
<!ELEMENT scenario (scenname, scenversion, sceninst, scentype, scenstarturl,
scenstartmod, scentexts, components)>
<!ELEMENT scenname (#PCDATA)>
<!ELEMENT scenversion (#PCDATA)>
<!ELEMENT sceninst (#PCDATA)>
<!ELEMENT scentype (#PCDATA)>
<!ELEMENT scenstarturl (#PCDATA)>
<!ELEMENT scenstartmod (#PCDATA)>
<!ELEMENT scentexts (scentext)>
<!ELEMENT scentext (scenlangu, scendesc)>
<!ELEMENT scenlangu (#PCDATA)>
<!ELEMENT scendesc (#PCDATA)>
<!ELEMENT components (component+)>
<!ELEMENT component (compname, compversion, comptype, comptexts, properties)>
<!ELEMENT compname (#PCDATA)>
<!ELEMENT compversion (#PCDATA)>
<!ELEMENT comptype (#PCDATA)>
<!ELEMENT comptexts (comptext)>
<!ELEMENT comptext (complangu, compdesc)>
<!ELEMENT complangu (#PCDATA)>
<!ELEMENT compdesc (#PCDATA)>
<!ELEMENT properties (property+)>
<!ELEMENT property (propname, propvalue)>
<!ELEMENT propname (#PCDATA)>
<!ELEMENT propversion (#PCDATA)>
```

If you upload the XML-based format using transaction GRMG, the data is stored in the specified tables. To clarify the expected data types, the tables show the mapping of Tag name to Field name.You can modify scenario customizing by changing the XML file and repeating the upload.

***GRMG_CONTROL*: This table contains administration information for controlling the GRMG Framework.**

Tag name	Field name	Key	Data type	Semantic
<grmgruns>	GRMG_RUNS	X	CHAR(1)	Flag to check whether GRMG Framework is running or not. Set to 'X' if you wish GRMG scenarios to be executed.
<runlog>	RUN_LOG	X	CHAR(1)	Flag to check whether run time logging is switched on or not. Should be set to <space> unless you are testing or debugging a scenario.
<errorlog>	ERROR_LOG	X	CHAR(1)	Flag to check whether error logging is switched on or not. Should be set to <space> unless you are testing or debugging a scenario.

***GRMG_SCENARIOS:* This table defines each scenario that is to be run and contains administrative information on each scenario.**

Tag name	Field name	Key	Data type	Semantic
<scenname>	SCEN_NAME	X	CHAR(8)	Technical name of the GRMG Application / Test Request
<scenversion>	SCEN_VERSION	X	NUMC(3)	Version of the GRMG Application / Test Request (support of more than one version of one GRMG Application / Test Request). The version must be greater than 0.

Tag name	Field name	Key	Data type	Semantic
<sceninst>	SCEN_INST	X	NUMC(3)	Instance of the GRMG Application / Test Request to support more than one running test scenarios within one solution. The instance must be greater than 0.
<scentype>	SCEN_TYPE		CHAR(10)	Type of the scenario. Valid values are: URL, if the scenario URL is held directly in SCEN_START_URL HRFC, if the scenario URL is stored in an SAP RFC destination of type HTTP (SAP or non-SAP component).
<scenstarturl>	SCEN_START_URL		CHAR(250)	For type URL, name of a start URL for the GRMG Application. For type HRFC, the name of the HTTP RFC destination to use for the scenario. The RFC destination contains the URL, and can also be used to store a logon user and password in secure form.
<scenstartmod>	SCEN_START_MOD		CHAR(50)	Name of a function module to run as the GRMG Application instead of POSTing to a URL. Not currently supported, should be set to a value such as 'Unknown'.

GRMG_SCENARIO_T: **This table contains a description of each scenario. The description is presented to users instead of the technical name of a scenario in the alert monitor. You MUST provide a description for each scenario.**

Tag name	Field name	Key	Data type	Semantic
<scenname>	SCEN_NAME	X	CHAR(8)	Name of the GRMG Application / Test Request
<scenversion>	SCEN_VERSION	X	NUMC(3)	Version of the GRMG Application / Test Request (support of more than one version of one GRMG Application / Test Request)
<scenlangu>	SCEN_LANGU	X	LANG(1)	Language for scenario description. The language is not used for selecting a scenario description. You should provide only one GRMG_SCENARIO_T entry per scenario. You should, if possible, always provide an English text (scen_langu E).
<scendesc>	SCEN_DESC		CHAR100	Language-dependent scenario description. The description is used as the name of the monitoring context that is created for the scenario in the monitoring architecture. Though the defined length is 100, you should limit the description to a maximum of 40 characters. Since the description is prefaced by Scenario Error in reporting scenario failures, you should make sure that all scenario descriptions are unique in the first 26 characters.

GRMG_COMPONENTS: **This table defines a component that is to be monitored. A scenario can request tests of one or more components. You must define at least one component.**

Tag name	Field name	Key	Data type	Semantic
<compname>	COMP_NAME	X	CHAR(8)	Technical name of a component
<compversion>	COMP_VERSION	X	NUMC(3)	Version of a component. Version must be > 0.
<comptype>	COMP_TYPE		CHAR(10)	Type of component. Not currently used. Should be set to 'Unknown' or some similar dummy value.

GRMG_COMPONENT_T: **This table contains a description of each component. You must provide a description of each component that you describe.**

Tag name	Field name	Key	Data type	Semantic
<compname>	COMP_NAME	X	CHAR(8)	technical name of the component
<compversion>	COMP_VERSION	X	NUMC(3)	Version of the component
<complangu>	COMP_LANGU	X	LANG(1)	Language for component description. You should provide only one description of each component, preferably in English (language key E).
<compdesc>	COMP_DESC		CHAR(100)	Language-dependent component description. The description is used as the name of the component in the monitoring architecture. Though the defined length is 100, you should limit the description to a maximum of 20 characters. The monitoring architecture allows up to 40 characters for the component name, but the name is suffixed with the hostname and component instance, which reduces the available length for the name.

GRMG_PROPERTIES: **This table contains the relations between scenario, components and properties for the single components. You must define at least one property for each combination of scenario and component, even if it is a dummy property that the GRMG application ignores.**

Tag name	Field name	Key	Data type	Semantic
<scenname>	SCEN_NAME	X	CHAR(8)	Technical name of the GRMG Application / Test Request
<scenversion>	SCEN_VERSION	X	NUMC(3)	Version of the GRMG Application / Test Request to support more than one software version of one GRMG Application / Test Request. Note that currently all versions of a scenario are executed.
<sceninst>	SCEN_INST	X	NUMC(3)	Instance of the GRMG Application / Test Request to support more than one running test scenarios within one solution
<compname>	COMP_NAME	X	CHAR(8)	Name of a tested component (technical component / application check)
<compversion>	COMP_VERSION	X	NUMC(3)	Version of a tested component (technical component / application check)

Tag name	Field name	Key	Data type	Semantic
<propname>	PROP_NAME	X	CHAR(40)	Name of a parameter to transfer in a GRMG request to a GRMG application
<proptype>	PROP_TYPE		CHAR(3)	Type of a parameter to transfer
<propvalue>	PROP_VALUE		CHAR(250)	Value of a parameter to transfer

2.2 Deleting GRMG Customizing

To delete customizing for a scenario, please use transaction SE16 to delete the table entries of the scenario from the tables listed in the previous section. Transaction GRMG offers a delete function, however, this function currently deletes only the monitoring trees in the Monitoring Architecture. The GRMG scenarios start to run again at the next system start or segment 'warmup' in the central server.

2.3 GRMG Request Processing

During the GRMG Request Processing, the information is read from the GRMG Customizing and is transferred in the XML-based request format. After this the start URL is called. The HTTP POST functionality is used to call the start URL. The XML-based request format is transferred within the HTML POST data. Below you can find an example showing the XML-based request format and the XML DTD.

```
<scenario>
          <scenname> .. </scenname>
          <scenversion> .. </scenversion>
          <sceninst> .. </sceninst>
               <component>
                         <compname> .. </compname>
                         <compversion> .. </compversion>
                              <property>
                                        <propname> .. </propname>
                                        <propvalue> .. </propvalue>
                              </property>
                              …
               </component>
               …
</scenario>

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT scenario (scenname, scenversion, sceninst, component+)>
<!ELEMENT scenname (#PCDATA)>
<!ELEMENT scenversion (#PCDATA)>
<!ELEMENT sceninst (#PCDATA)>
<!ELEMENT component (compname, compversion, property+)>
<!ELEMENT compname (#PCDATA)>
<!ELEMENT compversion (#PCDATA)>
<!ELEMENT property (propname, propvalue)>
<!ELEMENT propname (#PCDATA)>
<!ELEMENT propvalue (#PCDATA)>
```

2.4 GRMG Response Processing

After conducting any functionality testing that it is supposed to do, a GRMG Application must generate an XML document that conforms to the GRMG Response DTD. The GRMG Application then sends this response back to the GRMG Framework in the SAP System that is doing the monitoring. The GRMG Framework receives the information and interprets it, displaying the results in a CCMS monitoring tree.

A sample response and the DTD for the response format are shown below. Note that GRMG does not validate a response against a DTD. However, the parser is programmed to expect a DTD-compliant document, and parsing errors may be reported if a response is not compliant with the DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario>
          <scenname> .. </scenname>
          <scenversion> .. </scenversion>
          <sceninst> .. </sceninst>
               <component>
                    <compname> .. </compname>
                    <compversion> .. </compversion>
                    <comphost> .. </comphost>
                    <compinst> .. </compinst>
                    <messages>
                         <message>
                              <messalert> .. </messalert>
                              <messseverity> .. </messseverity>
                              <messarea> .. </messarea>
                              <messnumber> .. </messnumber>
                              <messparam1> .. </messparam1>
                              <messparam2> .. </messparam2>
                              <messparam3> .. </messparam3>
                              <messparam4> .. </messparam4>
                              <messtext> .. </messtext>
                         </message>
                         …
                    </messages>
               </component>

               …

</scenario>

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT scenario (scenname, scenversion, sceninst, component*)>
<!ELEMENT scenname (#PCDATA)>
<!ELEMENT scenversion (#PCDATA)>
<!ELEMENT sceninst (#PCDATA)>
<!ELEMENT component (compname, compversion, comphost, compinst, messages)>
<!ELEMENT compname (#PCDATA)>
<!ELEMENT compversion (#PCDATA)>
<!ELEMENT comphost (#PCDATA)>
<!ELEMENT compinst (#PCDATA)>
<!ELEMENT messages (message*)>
<!ELEMENT message (messalert, messseverity, ((messarea,messnumber, messpara1,
messpara2, messpara3, messpara4) | messtext))>
<!ELEMENT messalert (#PCDATA)>
<!ELEMENT messseverity (#PCDATA)>
<!ELEMENT messarea (#PCDATA)>
<!ELEMENT messnumber (#PCDATA)>
<!ELEMENT messpara1 (#PCDATA)>
<!ELEMENT messpara2 (#PCDATA)>
<!ELEMENT messpara3 (#PCDATA)>
<!ELEMENT messpara4 (#PCDATA)>
<!ELEMENT messtext (#PCDATA)>
```

Tag name	ABAP Data type	Semantic
<scenario>	No ABAP data type	Tag to summarize the scenario
<scenarioname>	CHAR(8)	Name of the GRMG Application / Test Request
<scenarioversion>	NUMC(3)	Version of the GRMG Application / Test Request to support more than one software version of one GRMG Application / Test Request
<scenarioinst>	NUMC(3)	Instance of the GRMG Application / Test Request to support more than one running test scenario within one solution
<component>	No ABAP data type	Tag to summarize the component
<compname>	CHAR(8)	Name of a tested component (technical component / application check)
<compversion>	NUMC(3)	Version of a tested component (technical component / application check)
<comphost>	CHAR(40)	Host where the component is running
<compinst>	CHAR(40)	Scalibility unit where the component is running (cardinality of the tested component, should more than one be tested by the GRMG application). Note that the compinst is not provided in the GRMG request but must be set by the GRMG application. You should always set the compinst to the correct value (1, if only one instance of a component is being tested). If it is not set in the response, then the GRMG Framework sets it to the value 'Unknown'. Since the compinst appears in the Monitoring Architecture, it is better to avoid having the default value 'Unknown' used.
<messages>	No ABAP data type	Tag to summarize messages
<message>	No ABAP data type	Tag to summarize a single message
<messalert>	CHAR(8)	Alert of a message (OKAY, ERROR), A message with the value OKAY is reported as 'GREEN' in the Monitoring Architecture. An ERROR message generates a red alert in the Monitoring Architecture
<messseverity>	NUMC(3)	Message severity (000 – 255). The severity is used to 'weight' a message within an alert level, in this case, the alert level 'red'. A higher number indicates that the problem is more severe and affects the propagation of the alert in the Alert Monitor (transaction RZ20). The default in the Monitoring Architecture is 50.
<messarea>	CHAR(20)	Message area. messarea and messnumber identify an SAP T100 message that should be reported into the Monitoring Architecture. If the required message area is not installed in the monitoring system, then you can report the message text in the messtext element.
<messnumber>	NUMC(3)	Message number (000 – 999). Number of a T100 message to report into the Monitoring Architecture.
<messpara1>	CHAR(120)	Message parameter 1. Parameter for a T100 message, if applicable.
<messpara2>	CHAR(120)	Message parameter 2. Parameter for a T100 message, if applicable.
<messpara3>	CHAR(120)	Message parameter 3. Parameter for a T100 message, if applicable.

Tag name	ABAP Data type	Semantic
<messpara4>	CHAR(120)	Message parameter 4. Parameter for a T100 message, if applicable.
<messtext>	CHAR(50)	Message text. You must in any case return a message text element, even if you cite a T100 message. Leave the element empty if you wish to have the T100 message reported. If a text is present, then the GRMG infrastructure reports the text and not the T100 message.

To display a message within the CCMS monitoring tree, you can use the normal ABAP message (message area, message number and message parameter 1 until 4) or a message text. We recommend that you work with ABAP messages.

3 GRMG Application

If an SAP application or other application is to be monitored by the GRMG, then an appropriate GRMG Application has to be implemented by the application development department. The GRMG Application receives the GRMG Request sent by the GRMG Framework including the transferred parameters. The GRMG Application then executes the checks to verify the availability of technical components or a business process step. The results of the checks are combined into the XML-based GRMG Response and are returned to the GRMG Framework.

To support the parsing of parameters from the GRMG Request and the creation of the GRMG Response we provide a set of application-independent Java classes and for the ABAP world a Web/AS service and handler class (as of SAP_BASIS Release 6.20).

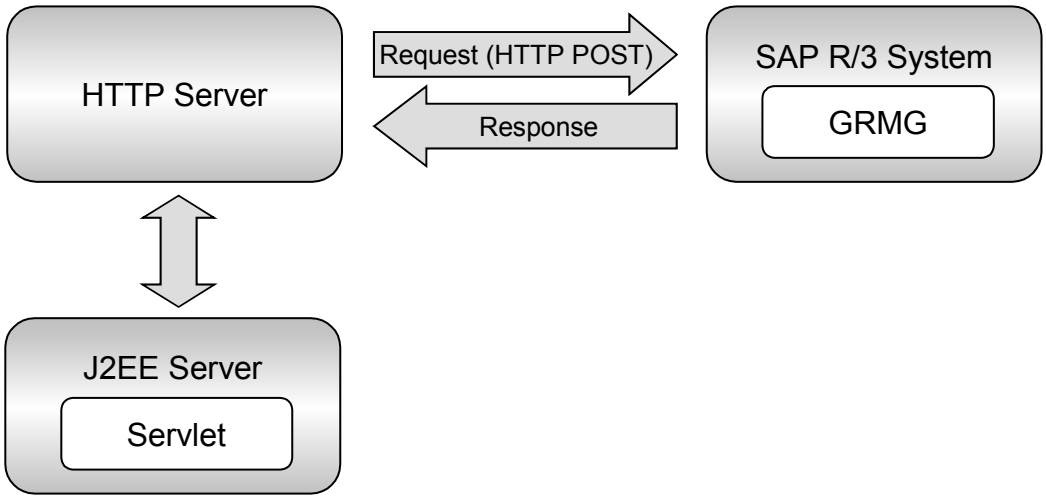
3.1 Java Helper Classes

The predefined Java classes, make it possible for a software developer easily to implement his or her own GRMG Application in Java. The following predefined classes can be used by software developers of GRMG Applications:

Class name	Description
GrmgRequest	Container for GRMG request
GrmgResponse	Container for GRMG response
GrmgScenario	Container for GRMG scenario
GrmgComponent	Container for GRMG component
GrmgProperty	Container for GRMG property
GrmgMessage	Container for GRMG message

These Java classes are available on request to one of the contact persons listed at the end of this document from SAP. They will also be integrated in the development (nightly build) and delivery environment by the middle of October 2002.

Consider the following GRMG test scenario: The GRMG Framework sends a GRMG Request to the HTTP server. The GRMG Request is transferred to the J2EE Server. The J2EE server with a deployed GRMG Application (JSP / Servlet with check functionality) responds with the appropriate GRMG Response.



The following XML document is an example of a GRMG Request (transfer of parameters):

```
<scenario>
          <scenname>tst</scenname>
          <scenversion>010</scenversion>
          <sceninst>100</sceninst>
               <component>
                    <compname>If web page exists</compname>
                    <compversion>001</compversion>
                         <property>
                              <propname>url</propname>

          <propvalue>http://localhost/index_test.html</propvalue>
                         </property>
               </component>
               <component>
                    <compname>Test connection to SAP R/3 system</compname>
                    <compversion>001</compversion>
                         <property>
                              <propname>SAPClient</propname>
                              <propvalue>ss</propvalue>
                         </property>
                         <property>
                              <propname>userid</propname>
                              <propvalue>KOJEVNIKOV</propvalue>
                         </property>
                         <property>
                              <propname>password</propname>
                              <propvalue>tstpw</propvalue>
                         </property>
                         <property>
                              <propname>language</propname>
                              <propvalue>DE</propvalue>
                         </property>
                         <property>
                              <propname>host_name</propname>
                              <propvalue>host</propvalue>
                         </property>
                         <property>
                              <propname> system_number </propname>
                              <propvalue>DE</propvalue>
                         </property>
               </component>
</scenario>
```

In order to interpret the GRMG Request, the GRMG Application must process HttpServletRequest and parse the InputStream in accordance with XML syntax. It can then proceed with functional checking tasks. Finally, it must generate the GRMG Response as an XML document. All of these tasks are simplified with the help of the predefined JAVA classes described above.

```
public void doPost(HttpServletRequest req , HttpServletResponse res)
          Throws ServletException, IOException
{
          ServletOutputStream out = res.getOutputStream();
          // Get output stream //of the servlet
          InputStream is=req.getInputStream();
          // Get input stream of servlet
          try
          {
               GrmgRequest ix=new GrmgRequest(is);
               // Create new GrmgRequest object  from
               //serlvlet's input stream and parse XML structure
               GrmgScenario sc = ix.getScenario();
               // Get GrmgScenario object from GrmgRequest
               processing(sc);
               /* You must include your own code here to process
                  the parameters passed by XML document and
                  to carry out any testing that is required */
               GrmgResponse ox=new GrmgResponse(sc);
               // Create new  GrmgResponse object using GrmgScenario
               //object as parameter
               ByteArrayOutputStream bytearray=ox.getOutput();
               // Get XML document as result of processing
               bytearray.writeTo(out);
               // give back result to servlet's output stream

          }
          catch (Exception e)
          {
               // Exception handling
          }
}
```

As shown, the software developer of the GRMG Application must add own coding at shaded area. It could look like the following example (in accordance with the example XML document above):

```
void processing(GrmgScenario sc) throws Exception
{
          try
          {
               URL conn = new URL(sc.getComponentByName("If web page exists").
               getPropertyByName("url").getValue());
               // getting URL
               HttpURLConnection hcon = (HttpURLConnection)
               conn.openConnection();
               // trying to connect to URL
               if(hcon.getResponseCode() == HttpURLConnection.HTTP_NOT_FOUND)
               //checking response from connection
                    sc.getComponentByName("If web page exists").addMessage().
                    setMessageParameters("ERROR","255","RT","001","","","","",
                    "page not found");
               // Page not found and we generate Message for GRMG
               else
                    sc.getComponentByName("If web page exists").addMessage().
                    setMessageParameters("OKAY","000","RT","003","","","","",
                    "relatively okay (for test)");
                    // OKAY and we generate Message for GRMG
          }
```

```
        catch(Exception e)
        {
            sc.getComponentByName("If web page exists").addMessage().
            SetMessageParameters("ERROR","255","RT","002",e.getMessage(),
            "","","",e.getMessage());
        }
        try
        {
            JCO.createClient(sc.getComponentByName("Test connection to R/3
            system").getPropertyByName("SAPClient").getValue();
            sc.getComponentByName("Test connection to R/3
            system").getPropertyByName("userid").getValue();
            sc.getComponentByName("Test connection to R/3
            system").getPropertyByName("password").getValue();
            sc.getComponentByName("Test connection to R/3
            system").getPropertyByName("language").getValue();
            sc.getComponentByName("Test connection to R/3
            system").getPropertyByName("host_name").getValue();
            sc.getComponentByName("Test connection to R/3
            system").getPropertyByName("system_number").getValue());
            // trying to connect to R/3 system
            sc.getComponentByName("Test connection to R/3 system").
            addMessage().setMessageParameters("OKAY","001","RT","004",
            "","","","","Connected!");
            //Everything is fine and we generate Message for GRMG

        }
        catch(Exception e)
        {
            // Exception handling
        }
}
```

The following deals with the shaded lines in more detail. Here we attempt to access the components using their names and to add messages that are meaningful for the GRMG Framework.

The parameters for the method setMessageParameters are:

- Result in the form of a string , that is, one of following values: OKAY or ERROR.
- Message severity – a weighting of an error message according to its severity, from 0 (not severe) to 255 (very severe). The default severity in the Monitoring Architecture is 50.
- A name of an SAP T100 message ID in the SAP System that is doing the monitoring
- Message number in the message class
- Parameter No 1 for the message
- Parameter No 2 for the message
- Parameter No 3 for the message
- Parameter No 4 for the message
- Message text

This means that in order to propagate messages for the GRMG Framework you have to specify a message area and a message number within the SAP Web AS System in which the GRMG Framework is located or to define a static message text.

3.2 ABAP GRMG Applications

As we have noted above, SAP delivers a ready-to-use GRMG application in Web/Application Servers as of SAP_BASIS Release 6.20 Support Package 11. The ABAP application ensures that GRMG can be used for all component-level availability testing, at least for components running on Web/AS Release 6.20.

The standard ABAP GRMG application offers less flexibility than Java implementations. There are two ways to use the standard application:

- You can have the application run a specified function module which can in turn carry out whatever testing is necessary.

  The function module can either implement the interface modeled in function module GRMG_ABAP_REFERENCE_TOOL (function group GRMG). The ABAP GRMG application can also run and interpret the result of any function module that requires no parameters and which returns a non-zero return code in case of an error.

- You can send a GRMG request to the ABAP GRMG application without specifying any function module to run. In this case, the GRMG application essentially answers (or fails to answer) a ping against the application server that handles the request.

If you have testing requirements which cannot be satisfied by the standard application, then you can copy and modify the handler classes CL_CCMS_GRMG_APPLICATION and CX_CCMS_GRMG_APPLICATION_ERROR.

Customizing for an ABAP scenario is exactly as described above. The only difference lies in the following optional properties which you can use only in the ABAP scenario. You should maintain these properties in an XML customizing file just as you would for any other GRMG properties. Like other GRMG properties, they are inserted into the GRMG_PROPERTIES table.

Property name	ABAP Data type	Semantic
funcmod	String	Name of a function module that is to be executed by the GRMG application. If the function module is active in the system in which the application is running, then it is executed. If not, it is skipped and an error message is returned in the GRMG response.
grmg_compliant	String	Specifies whether the function module specified in the funcmod property complies with the interface defined in GRMG_ABAP_REFERENCE_TOOL in function group GRMG. Value YES or yes means that the function module is compliant with the interface. Any other value or omission of the parameter means that the function module is not compliant. In this case, the function module may not require any parameters and may not return any messages. A return code of 0 from the function module is interpreted by the GRMG application as condition OKAY, any other return code causes the GRMG Response to return a condition of ERROR.
dest	String	The RFC destination at which the function module specified with property funcmod is to be executed.
parname[1-5]	String	Names of parameters which are passed to a GRMG-compliant function module when it is called.
parvalue[1-5]	String	Values for the parameters named in any parname<n> properties.

Depending upon your support package level, you may need to define a service in a Release 6.20 Web/AS for calls to an ABAP GRMG Application. Because of a transport error, the standard service is not included in SAP_BASIS Support Package 11.

The specifications to make in defining such a service in transaction SICF are as follows:

Field name	Value
Path	/default_host/sap/bc/ccms/monitoring/GRMG_APP
ICF handler (in Handler list tab)	CL_CCMS_GRMG_APPLICATION
User and password	Either • Leave these fields blank and supply a logon user and password by means of an RFC destination of type HTTP in the monitoring system or • Enter the logon data for a service user with limited authorizations such as the CCMS CSMREG user.

4 Running GRMG Scenarios

Any GRMG scenarios that have been uploaded into the customizing tables are run automatically whenever the monitoring system is restarted or whenever the monitoring segment of the central instance (the instance that offers the Enqueue Service) of the system is set to 'Warmup' status (see the CCMS documentation for more details).

By default, all GRMG monitoring trees are created in the monitoring shared memory segment of the central server.

You can also start scenarios by hand from the transaction GRMG. If you do so, make sure that you are logged in on the central server. Otherwise, execution of any scenarios which have already run automatically will end with scenario errors. Since GRMG monitoring trees have a 'system-wide' validity, they cannot be created in the monitoring segments of other application servers of the monitoring system.

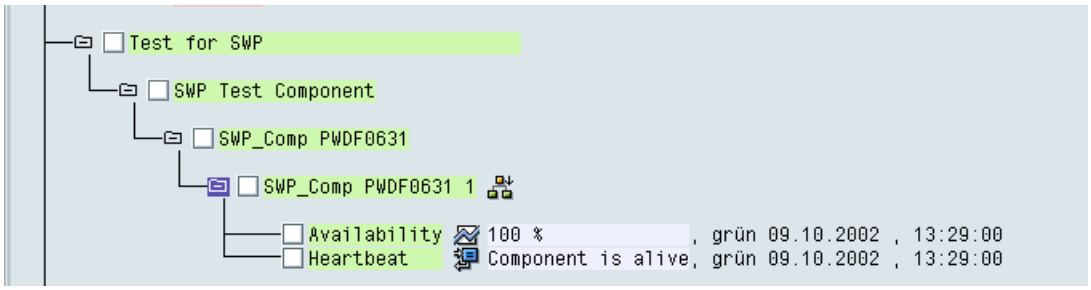
5 GRMG Error Reporting

GRMG is designed with a 'testing agent' concept in mind. That is, ideally, the GRMG application runs as an agent apart from the components or applications on which it reports. The extra level of indirection between the GRMG framework and the tested components or applications allows GRMG to distinguish between scenario failures – any scenario for which no response or an invalid response was received - and component errors, actual problems in the components that are being monitored, which problems are returned to the GRMG Framework in a valid GRMG Response. A scenario failure such as a communication error with an agent does not necessarily mean that the tested components are also unavailable, so the distinction between scenario failure and component error is a useful one.

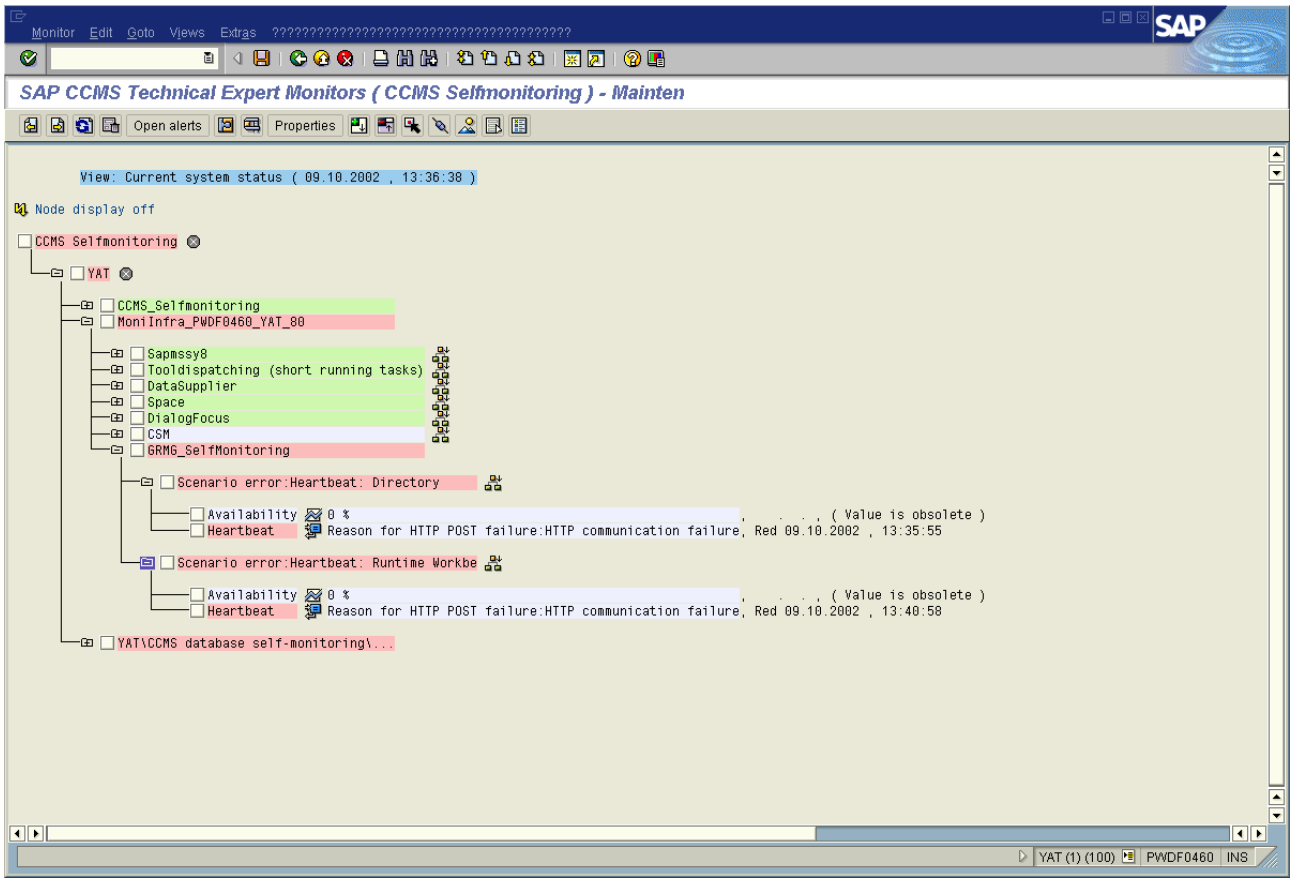
Because of this agent-based design and also because, in the event of a scenario error, the GRMG framework cannot be sure that it can identify all of the monitoring trees affected by the failure, GRMG reports all scenario failures separately from the monitoring tree for component errors. This has consequences for the design of a GRMG monitor in the RZ20 alert monitor and also for your error analysis as you design and maintain GRMG scenarios.

The agent-based design of GRMG does not mean, by the way, that you cannot implant a GRMG application in a component or application that you wish to test. However, in such a case, you should document for users the fact that a scenario failure has the same meaning as a component error: the monitored component or application is not available for use.

Component errors reported by a GRMG application are reported in a monitoring tree anchored by a context that uses as its name the description of the scenario, as in this screen shot.



Scenario errors are reported in the CCMS Self-Monitoring for each application server in the monitoring system on which a scenario was executed. In the screen shot, you can see that communication errors have been reported for two GRMG scenarios. The communication errors indicate that the J2EE servers (in this case) on which the GRMG applications were running were not up at the time the scenario was tried.



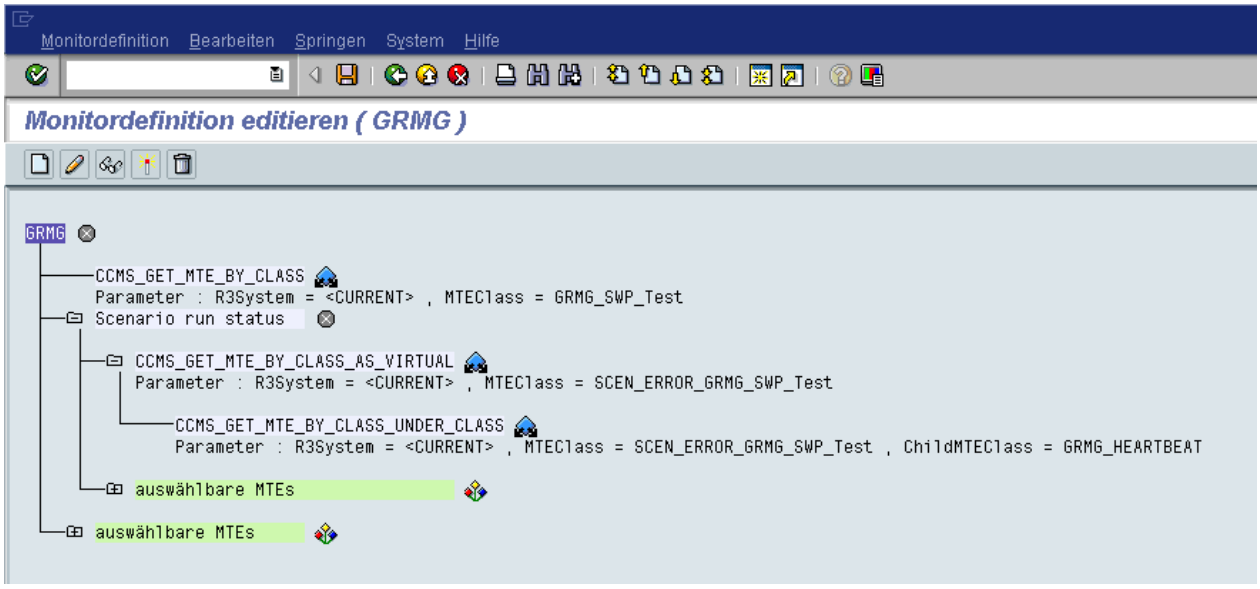
GRMG continues to repeat a failed scenario until it is again active or until you delete the scenario from the customizing tables. Even if the scenario has never run successfully, it is triggered by the Scenario Error monitoring tree.

GRMG also records scenario runs and errors in two database tables, GRMG_RUN_LOG and GRMG_ERROR_LOG, if the corresponding fields in the GRMG_CONTROL table have been set. The error table returns essentially the same messages that you see in the scenario error monitoring tree.

You should turn on this error logging only if you are developing or debugging a scenario. Otherwise, the tables may grow quite large.

6 GRMG Sample Monitor Definition

Here is a sample monitor definition for a GRMG scenario.



The rules in the definition do the following:

- CCMS_GET_MTE_BY_CLASS selects the context of the monitoring tree in which component availability and component errors and status are reported. The MTEs in this tree are provided with data when the scenario runs successfully (a valid response is received by the GRMG Framework, no matter whether the monitored component is available or not).

  As you can see, the MTE class of the monitoring context of a GRMG scenario is formed by concatenating the string

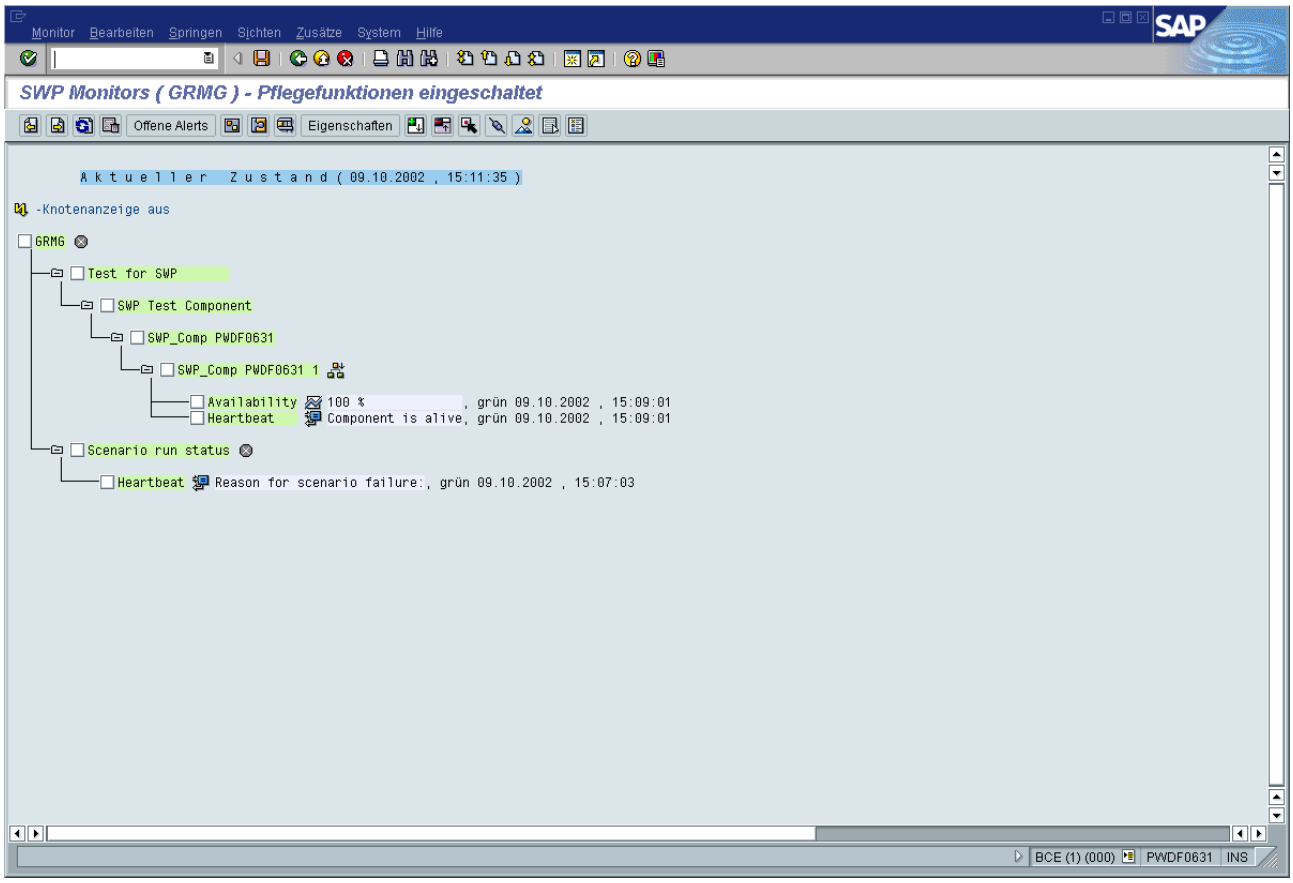
  GRMG_

  with the technical name of the scenario. In this case that is

  SWP_Test.

- Under the Scenario run status virtual MTE, two rules select the message log in which scenario failures are reported. You can also include the scenario availability MTE, which reports on the percentage of time that the scenario has run correctly.

The generated monitor looks like this:



7 GRMG Enabling to-do-list

Task name	Explanation
Check the technical prerequisites for GRMG enabling	The GRMG Framework is available within a SAP Web AS as of release 6.20, Support Package 11. As of SAP Web AS 6.30, the latest version of the GRMG Framework is part of the standard SAP system and will be updated using standard Support Packages. SAP recommends that you include a dedicated Release 6.20 System as a central monitoring system in your IT landscape.
Check the application prerequisites for GRMG enabling	The GRMG test scenario for which the availability is to be checked must be described as a straight forward business case. Technical scenarios with different software components (especially if there is no active data provider for the software components involved) and Web based e-business scenarios are good candidates for GRMG enabling.
Specify GRMG Test Scenario – Concept	Before starting the implementation of the GRMG Application, the test scenario has to be clear. The slide “GRMG Test Scenario – Concept” (see above) can be used for this.
Specify GRMG Request / GRMG Customizing	To clarify the interface between the GRMG Framework and the GRMG Application (components and parameters per components), you must specify the XML-based request (see request format above) sent by the GRMG Framework to the GRMG Application. The GRMG Customizing file can be derived from the GRMG Request.
Specify GRMG Response / CCMS monitoring tree	To clarify the interface between the GRMG Application and the GRMG Framework (components and messages per components), you must specify the XML-based GRMG Response (see response format above) sent by the GRMG Application to the GRMG Framework. The CCMS monitoring tree can be derived from the GRMG Response. If the SAP Web AS-based messages are to be used, address desired messages to GRMG Framework development.

8 Contact persons

In case of questions or concerns please contact Stephen Pfeiffer (SAP Technology Development), Pavel Kojevnikov (SAP Mission Critical Support Technology), or Janko Budzisch (SAP Mission Critical Support Technology).