HealthCheck

In case of questions and comments about this document please feel free and contact

Phone: +49 (711) 4102 1541

Email: info@rockit.consulting

Detailed information about the document

Status of the document?

|  |  |  |
| --- | --- | --- |
| Status Details | | |
| Document Status | Date | Name |
| Possible characteristics:  In progress  In approval  Approved | Since when?  Since when?  Date of approval | Who?  Who?  Who? |
| In progress | 03/08/2018 | rockIT.consulting: J. Boehm |

Who wrote or changed this document?

|  |  |  |  |
| --- | --- | --- | --- |
| Version Control | | | |
| Version | Changes | Author(s) | Date |
| 1.0 | Development | J. Boehm | 08.03.2018 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Who was involved in the approval or the review (technical design, review, approval etc.)?

|  |  |  |
| --- | --- | --- |
| Recipient | | |
| Name | Company | Copies |
| Y. Dmukh | rock IT Consulting GmbH |  |
|  |  |  |

Inhalt

[Application “HealthCheck” 4](#_Toc508799810)

[Message Flows 4](#_Toc508799811)

[HealthChecker.msgflow 4](#_Toc508799812)

[HTTPHealthReport.msqflow 5](#_Toc508799813)

[XML-Files 6](#_Toc508799814)

[Exposure of the XML-file containing the information of the interfaces 6](#_Toc508799815)

[Exposure of the XML-file containing the results of the analysis 7](#_Toc508799816)

[Deployment and Operations 7](#_Toc508799817)

[Requirements 7](#_Toc508799818)

[MQ 8](#_Toc508799819)

[Application configuration 8](#_Toc508799820)

[Deployment 8](#_Toc508799821)

[Step-by-Step 8](#_Toc508799822)

[Example for HealthCheck.properties file 8](#_Toc508799823)

[Operation 9](#_Toc508799824)

[Automatic 9](#_Toc508799825)

[On Demand 9](#_Toc508799826)

[Logging 9](#_Toc508799827)

[PIA Logging Properties File Content Example 9](#_Toc508799828)

[Info Log Example 12](#_Toc508799829)

[Error Log Example 12](#_Toc508799830)

# Application “HealthCheck”

This application monitors external interfaces on the basis of an xml-file that describes the interfaces. The results of the monitoring are then provided in a report file. This report file is also built in xml format. The second component of this application display the report file in an internet browser.

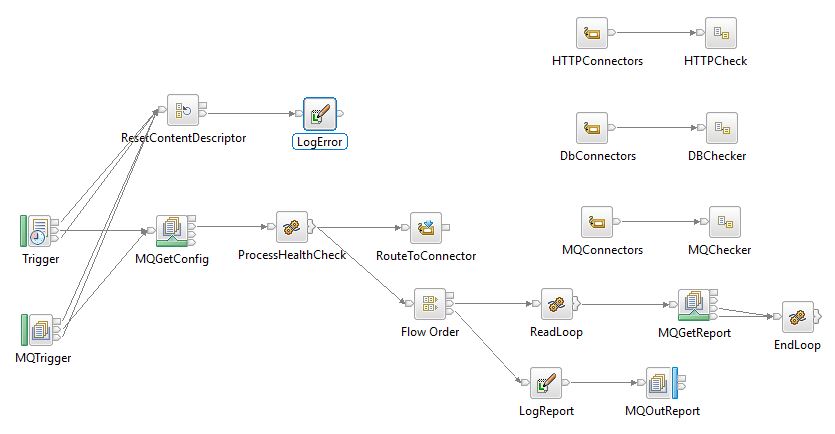
## Message Flows

The application consists of two message flows. The first flow monitors the external interfaces on the basis of an xml-file and writes the results in a report file. The second flow makes the report file available in an internet browser.

### HealthChecker.msgflow

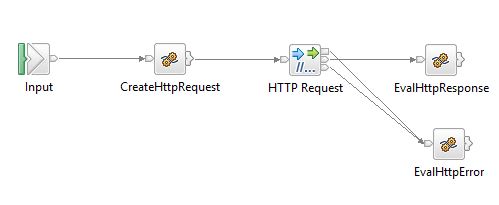
The “HealthChecker” flow is designed to work automatically. However, it can also be triggered to a start message. Responsible for the automatic activation of the flow is a Timeout Notification Node which triggers the flow every half hour. The flow can also be activated through a start message that is placed in HEALTH.TRIGGER Queue.

When the flow receives a trigger, it reads out the interface configuration of the xml file that is stored in the HEALTH.CONFIG Queue. With this information, the flow builds requests that are sent to these interfaces (web sides, databases and queues). Afterwards, the flow collects the answers from these interfaces and builds a report file. Before the report file is placed in the HEALTH.REPORT Queue, outdated message in the HEALTH.REPORT are deleted.



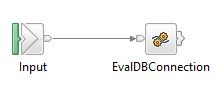
The “HealthChecker” flow uses for the evaluation of the interfaces the following three subflows.

#### HTTPChecker.subflow



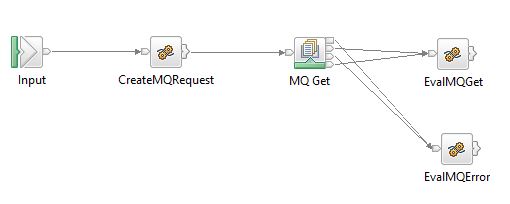
The “HTTPChecker” subflow checks the availability of a web site. There are three responses: side available, side not available and server not available. All the results are documented in the environment.

#### DBChecker.subflow



The “DBChecker” subflow controls a database connection. The flow tries to make a connection a table in a database schema. The outcome is stored in the environment and is available for processing in the main flow.

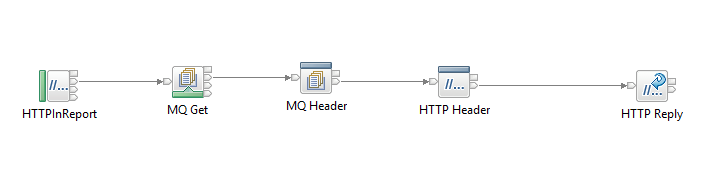
#### MQChecker.subflow



The “MQChecker” subflow tests the availability of a queue. The finding is again stored in the environment and afterwards processed from the main flow.

### HTTPHealthReport.msqflow

The “HTTPHealthReport” flow is started through a URL request. Subsequently, the flow reads out the report file which stands in the HEALTH.REPORT queue. Finally, the information gathered from the report file is depicted in xml structure in a browser.



## XML-Files

The application works with two xml files. The first file, contains the configuration of the interfaces. The second file is generated through the flow and contains the results from the requests.

### Exposure of the XML-file containing the information of the interfaces

<?xml version=**"1.0"** encoding=**"UTF-8"**?>

<Connectors xmlns:xsi=**"http://www.w3.org/2001/XMLSchema-instance"** xsi:noNamespaceSchemaLocation=**"HealthCheckConfiguration.xsd"**>

<HTTPConnectors>

<HTTPConnector>

<Id>**http1**</Id>

<RequestURL>**http://localhost:8040/test/mock1**</RequestURL>

<RequestLine>

<Method>**GET**</Method>

</RequestLine>

</HTTPConnector>

<HTTPConnector>

<Id>**http2**</Id>

<RequestURL>**http://localhost:8040/test/mock2**</RequestURL>

</HTTPConnector>

</HTTPConnectors>

<DbConnectors>

<DbConnector>

<Id>**mbvm1**</Id>

<ODBCDatasource>**MBVM01**</ODBCDatasource>

<DBSchema>**MBVM01**</DBSchema>

<DBTable>**MBVM001\_FZB\_MAPPING\_CONFIG**</DBTable>

</DbConnector>

<DbConnector>

<Id>**mbvm2**</Id>

<ODBCDatasource>**MBVXAI01**</ODBCDatasource>

<DBSchema>**CRM**</DBSchema>

<DBTable>**ABBATCH**</DBTable>

</DbConnector>

</DbConnectors>

<MQConnectors>

<MQConnector>

<Id>**Q1**</Id>

<QueueName>**IAP.EAI\_SMPL.U.EJBACT.ERR**</QueueName>

</MQConnector>

<MQConnector>

<Id>**Q2**</Id>

<QueueName>**IAP.EAI\_SMPL.U.EJBACT.REQ**</QueueName>

</MQConnector>

</MQConnectors>

</Connectors>

### Exposure of the XML-file containing the results of the analysis

<?xml version=**"1.0"** encoding=**"UTF-8"**?>

<HealthCheckResponse xmlns:xsi=**"http://www.w3.org/2001/XMLSchema-instance"** xsi:noNamespaceSchemaLocation=**"HealthCheckConfiguration.xsd"**>

<ResponseTimestamp>2018-03-16T14:13:57.734+01:00</ResponseTimestamp>

<Connectors>

<HTTPConnectors>

<HTTPConnector>

<Id>**http1**</Id>

<Status>**OK**</Status>

</HTTPConnector>

<HTTPConnector>

<Id>**http2**</Id>

<Status>**NOK**</Status>

<Error>**Service not available**</Error>

</HTTPConnector>

</HTTPConnectors>

<DbConnectors>

<DbConnector>

<Id>**mbvm1**</Id>

<Status>**OK**</Status>

</DbConnector>

<DbConnector>

<Id>**mbvm2**</Id>

<Status>**NOK**</Status>

<Error>**SQL172762...**</Error>

</DbConnector>

</DbConnectors>

<MQConnectors>

<MQConnector>

<Id>**Q1**</Id>

<Status>**OK**</Status>

</MQConnector>

<MQConnector>

<Id>**Q2**</Id>

<Status>**NOK**</Status>

<Error>**MQ234...**</Error>

</MQConnector>

</MQConnectors>

</Connectors>

</HealthCheckResponse>

# Deployment and Operations

Before the application can be deployed on the broker following requirements must be fulfilled.

## Requirements

### MQ

The following local queues need to be created on the queue manager which will host the application

* HEALTH.CONFIG – contains the application configuration
* HEALTH.REPORT – contains the current health check result
* HEALTH.TRIGGER – input queue for messages which trigger the health check execution

If you want to use your own queue names please set the corresponding values in the **HealthCheck.properties** file and override the bar file with this property-file (see Deployment).

### Application configuration

The XML configuration needs to be written to the HEALTH.CONFIG queue. This is required before deploying the application since the configuration is loaded immediately after the application start.

For an example of XML configuration see “Exposure of the XML-file containing the information of the interfaces”.

## Deployment

### Step-by-Step

|  |  |
| --- | --- |
| 1. | For the target environment the properties from the file **HealthCheck.properties** must be overridden:  *HTTPHealthReport#HTTPInReport.URLSpecifier* – URL path for the HealthCheck HTTP-Service  *HealthChecker#Trigger.timeoutInterval* – Time interval specifying how often the HealthCheck is triggered  *HealthChecker#ErrorTrace.filePath* – Full path of the error output file |
| 2. | For the HealthCheck application a bar file must be created e. g. HealthCheck\_v10.ba |
| 3. | The property values have to be applied to the bar file providing the new overridden bar file e. g. HealthCheck\_DEV.bar:  *mqsiapplybaroverride -b HealthCheck\_v10.bar -p HealthCheck.properties -r -o HealthCheck\_DEV.bar* |
| 4. | Please deploy the created bar file on the target execution group. |

### Example for HealthCheck.properties file

HTTPHealthReport#MQ Get.queueName = HEALTH.REPORT

HTTPHealthReport#HTTPInReport.URLSpecifier = /infra/healthcheck

HealthChecker#ReportQueue = HEALTH.REPORT

HealthChecker#MQGetConfig.queueName = HEALTH.CONFIG

HealthChecker#Trigger.timeoutInterval = 1800

HealthChecker#ErrorTrace.filePath = c:\errorTrace\healthCheckErrorTrace.txt

HealthChecker#MQTrigger.queueName = HEALTH.TRIGGER

## Operation

### Automatic

The application is designed to automatically generate report files. The interval is specified in the parameter HealthChecker#Trigger.timeoutInterval in the HealthCheck.properties file. By default the parameter is set to 1800ms which equates to 30 minutes.

### On Demand

However, the application can also be triggered on demand. For this case, a start message needs to be placed in the HEALTH.TRIGGER Queue. As soon the application receives this message, it collect the information and prepare a new report file.

## Logging

The “HealthCheck” application uses common logging. This logging is based in PIA logging. The logging is configured in a special pia logging properties file.

### PIA Logging Properties File Content Example



### Info Log Example



### Error Log Example

