
# ACTIVE-ACTIVE

# BOX MESSAGING HUB Stateful Arbitration

# Concept and Implementation

**Revision 1.0**

# Table of Content

# Table of Graphs

# 1 Concept

## 1.1 Introduction

BOX Messaging Hub Active-Active implementation (Active-Active) is a profound enhancement of past releases and supports current demands on high availability and instant payment with a dedicated configuration reflecting the architecture and, at its heart of Active-Active, the process of arbitration.

Resulting in a system sharing a (preferably) clustered database and backend applications and contrary to stateless systems, Active-Active has been implemented as a system keeping state (stateful system).

The present document aims at providing the reader with a theoretical approach to the BOX Messaging Hub 'Active-Active' and a detailed practical approach to the implementation of the Active-Active system. Due to the complexity of the system, prefacing preparations and a profound understanding of involved parts are fundamental to the success.

To reflect the approach mentioned, the document is divided in a conceptual and a practical part.

## 1.2 Active-Active

BOX Messaging Interface (BOX) now supports a stateful arbitrative concept with new enhancements, specified as Active-Active `(A-A)`, aiming to provide an increase of resilience and availability to fulfil customer requirements. With Active-Active in place, the BOX system provides an all-time available service, a better utilization of existing hardware and the support of WebSphere MQ Cluster Architecture.

BOX Active-Active increases the complexity of the system and needs therefore further attention to detail in implementation and configuration. The following graph gives an overview of the components of an Active-Active BOX system.

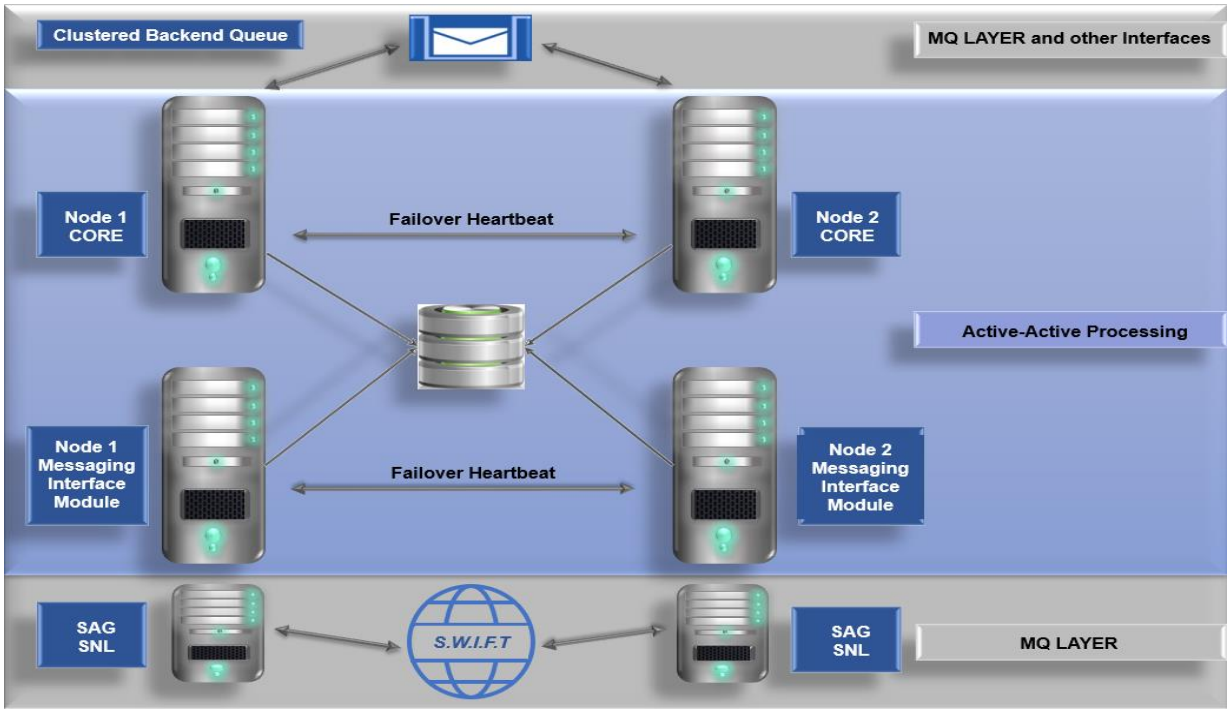



Figure 1 Overview of Active-Active in BOX

**Active-Active provides for**

- 24x7 Availability with planned downtimes
  - Not covering all Active-Updating scenarios (Continuous Availability)

- No service interruption if one or more components are failing
  - Processing of messages / files remains active automatic or operator-supported failover (configurable)
  - Combine Active-Active with traffic distribution options (BOX Messaging Gateway Modules and e.g. SwiftNet traffic distribution)

- Using a Single Database
  - Database Cluster to secure data (strongly recommended)
  - Storing Messages and configuration

- WebSphere MQ Cluster Support
  - MQ Cluster is only loosely coupled to Active-Active Implementation
    - Active-Active Setup should be combined with MQ Cluster (non-z/OS)
    - Active-Active Setup is also possible without MQ Cluster
    - MQ Cluster may be used without Active-Active Configuration

- Integration with existing BOX-Monitoring
  - Enhancements in BOX Domain Monitor concept

- Scaling & Resilience
  - Clustered Operation of multiple instances for all BOX components possible
  - Non-Active-Active configuration on appropriate hardware yields same performance as distributed installation
  - Active-Active configuration may increase complexity (architecture & operations)

Please note, SYSPLEX systems use a different concept and are already secure on OS- and Data-level. The BOX Messaging Hub Active-Active uses its own architecture to secure messaging operation.

## 1.3 General Overview

The following graph gives a general overview of the concept of Active-Active and how it can be implemented without and with the use of a clustered MQ

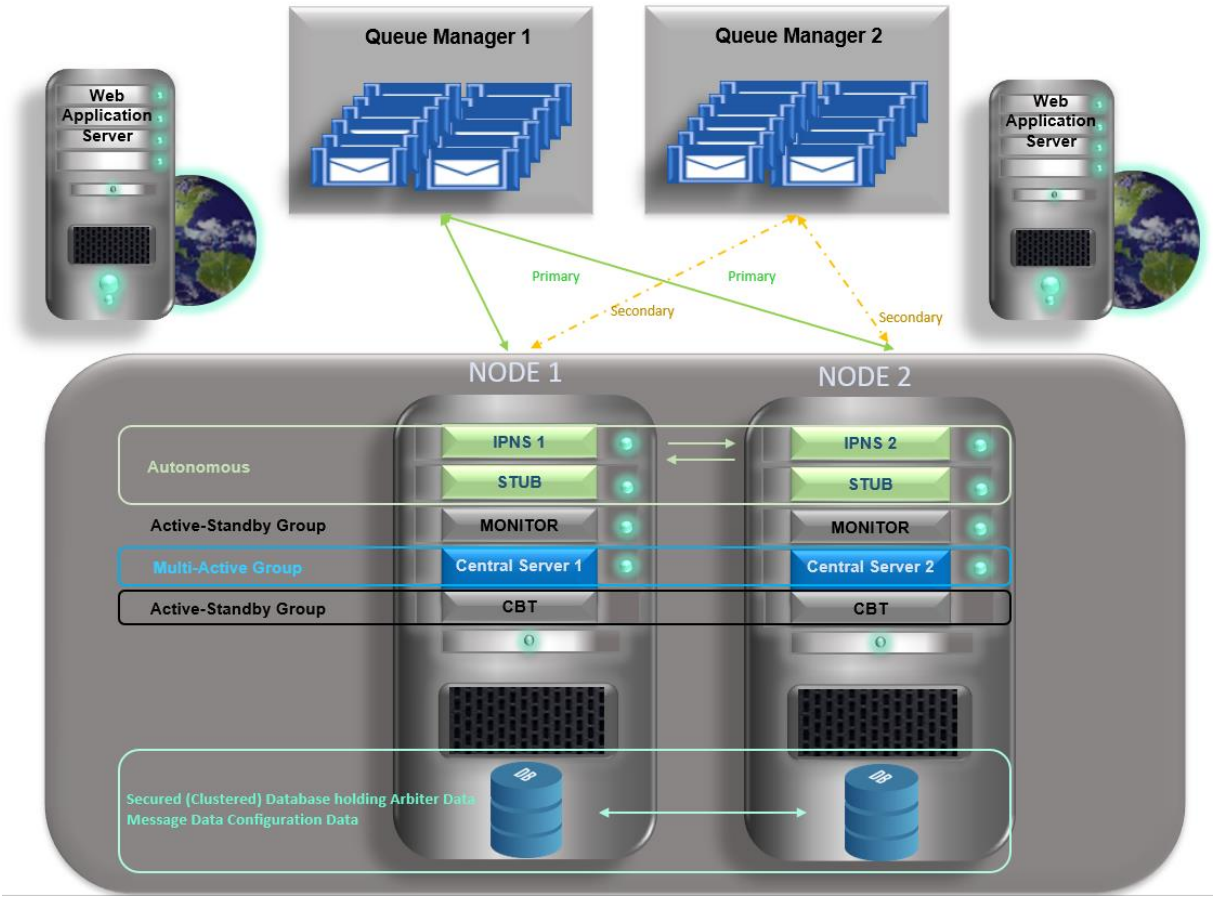



Figure 2 Detailed Active-Active Implementation in BOX (2 Nodes, 2 Queue Managers)

## 1.4 Module Groups and Instances

To implement and fulfill 'Active-Active' requirements, **module groups** are introduced. A module group combines logically linked modules into a group of either an active-active or an active-standby mode. A maximum of five instances form a Module Group, in the following chapters 1.4.1 and 1.4.2 described as Active-Standby and Active-Active.

### 1.4.1 Active-Standby

Active-Standby groups include module instances of the same logical instance (n number of it) with the same Module- and Service ID only. There will be only one active instance.

Instances belonging to the Active-Standby Group are the Monitor, Messaging (- and Communication) Gateways and Rendering Modules (DRM).

### 1.4.2 Active-Active

An Active-Active Module Group (one per domain only) comprises n number of modules, where each reflects one Instance, for the same purpose, but with different ModuleIDs. Several group members can be active. Active-Active Module Groups can only be used for the Central Server Module Type.

### 1.4.3 Standalone

To provide for a status for IPNS and STUB shown in the GUI, entries are generated in the arbitration table and a further category has been created, Standalone-Group.

Here, a module represents a group shown in the GUI and properties are no master and no takeover.

### 1.4.4 Table of Module-Types

| Mod-Type | Group-Type | Shared Data | Details |
|---|---|---|---|
| SERV | Active-Active | DB | one Group per BMH Domain |
| IPNS | Standalone | None (on each node) | Linked through IPNS-Cfg Independent Instances on each Node |
| STUB | Standalone | None (on each node) | Independent Instances on each Node |
| MON | Active-Standby | None / Shared-Drive + DB | Shared Drive for domaindb is optional |
| MGTW | Active-Standby | DB | Additional concept is Traffic Distribution |
| CGTW | Active-Standby | DB + Shared Drive | **Shared Drive is currently mandatory! RMA will be exluded** |
| DRM | Active-Standby | DB | Recovery through SERV-Formatter |

Table 1 Table of Module- and Mode-Types

### 1.4.5 Module Instances and Instance Numbers

An Instance Number for each installed module (IPNS. STUB, MON, SERV, CGTW, MGTW, DRM) has also been introduced, which is given as command line parameter, is optional and defaults to 1 when starting a module.

#### 1.4.5.1 Instance Number Changes

The instance number changes for some of the interfaces offered by each module instance. These include console-shared memory, command-pipe name or communication names used in intra-MPO TCP/Pipes communication. The **console-shared memory name** and **command-pipe name** now change to mmmmssii (upper case hexadecimal representations)

| Number Format | Instance |
|---|---|
| mmmm | ModuleID |
| ss | ServerID |
| ii | InstanceNumber |

Table 2 Module Instance ID Setup

**Please note the different communication interfaces:**

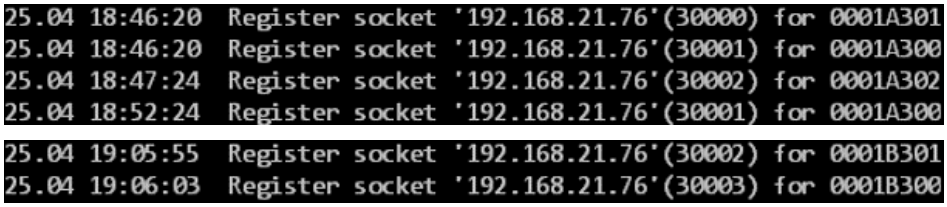



Figure 3 Console Output of Process List

#### 1.4.5.2 Instance number usage in tools

| Instance | Description |
|---|---|
| **mpo_cout** | Uses InstanceNumber, default is 1 |
| **mpo_shut** | Use InstanceNumber, default is 0 |
| **mpo_slog** | Use InstanceNumber, default is 0 |
| **mpo_srvsig** | Use InstanceNumber, default is 0. |
| **mpo_mcmd** | Use InstanceNumber, default is 0. |
| **mpo_mcmon** | Do not use InstanceNumber (implicit usage of 0) |
| **mpo_cmon** | Do not use InstanceNumber (implicit usage of 0) |

Table 3 Instance Number Usage

The instance number now is also used to generate the default configuration file and log files names when starting a module.

### 1.4.6 Web Application

Each of Multiple BOX Web-Applications (Load Balancer) connects to each Node.

All communication is done via the database, through which signalling must be established to all active modules on respective nodes. In a shared system, each web client must be uniquely identifiable. The parameter System.VMID in the configuration file ‘configuration.properties’ specifies the unique ID of the MP/O Java API. It is used to uniquely identify an instance of the MP/O Java API within the whole system.

If, for example, a web client is deployed on two different nodes connecting to each node, the configuration should be as such:

Web Client on Node 1

```
System.VMID=BOX-Client00000001
```

Web Client on Node 2

```
System.VMID=BOX-Client00000002
```

## 1.5 Arbitration

### 1.5.1 Principle of Arbitration

The basis of active-standby is the configuration of nodes in an arbitrative mode. Defined as the process of solving an argument between people by helping them to agree to an acceptable solution through an arbitrator, in technical terms, the **process of arbitration** describes the negotiation between modules to become active or remain in standby mode. Becoming active also includes also taking over the responsibility for the **Message Processing** of messages, where the OwningServer (CreationServer) has become inactive.

To support this architecture, the BOX module configuration has changed.

All Members of a Module Group jointly agree on their role and responsibilities in this group.

Some Module Group Members might no longer be able to participate in this process and other Members need to stand in. All Members acquire through the arbitration process an active or standby status. The active role is defined as Master and performs specific tasks.

All Arbitration Status Data are maintained in a table of the shared database.

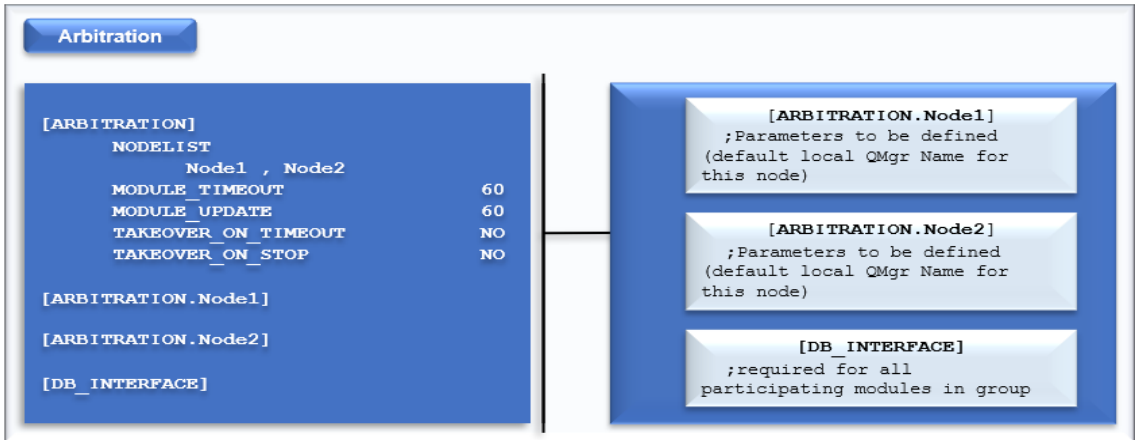



Figure 4 Module Arbitration Section

### 1.5.2 Arbitration in BOX

Status transmissions are an important part of the arbitration process and are assigned to specific key processing points. Without listing all available status checks, the main status parameters should be mentioned.

- Member status
- Master Role Assignment (Active Role assignment)
- Message Processing Map (Multi-Active Member signals processing activity for specific pending messages)
- Message Processor (Arbitration Takeover Result for pending messages of a specific member)

#### 1.5.2.1 Takeover Details

To avoid concurrent MPS Processing on different Central Server Instances (Data Integrity Protection), a **Takeover** process has been implemented supporting a stateful architecture, whereby another (Central) Server Module continues the MPS processing on behalf of the unavailable Module Group Member. This involves MPS requeuing of active messages, issuing delivery notification and response handling and processing of asynchronous, external responses, such as Embargo. Configuration allows automatic and operator-driven options for Takeover.

### 1.5.2.2 Modules Group Views

The following graphs represent the view on Module Groups of a small system within the BOX Web Client (user must be on Enterprise level).

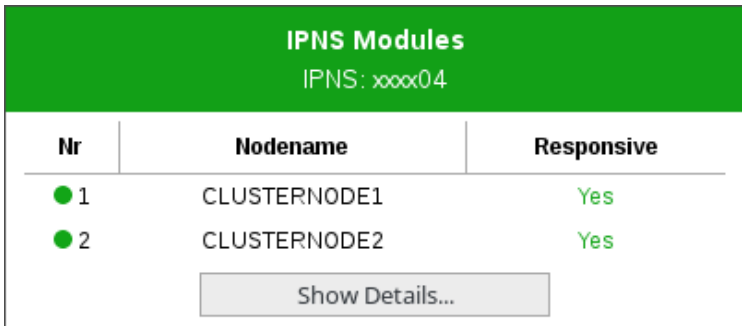



Figure 5 View Standalone Group IPNS

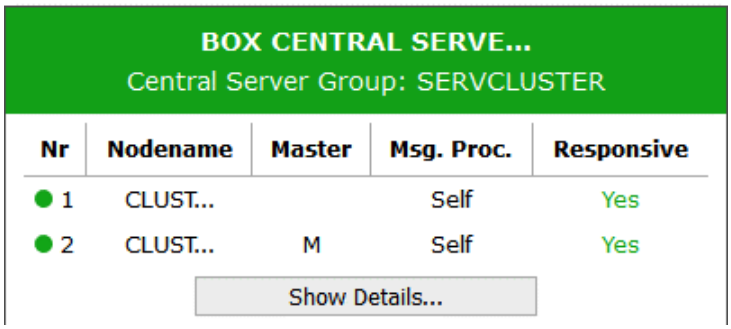



Figure 6 View Multi-Active Group Central Server

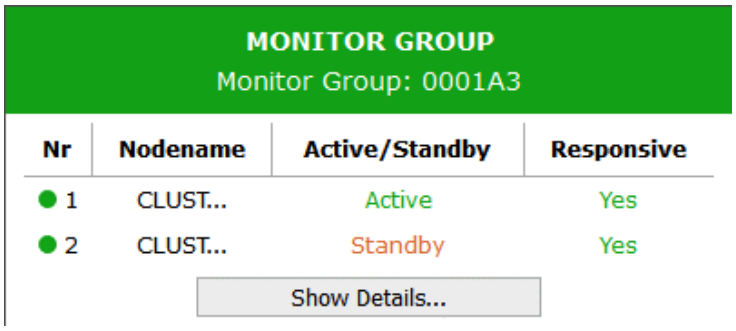



Figure 7 View Active-Standby Group Monitors

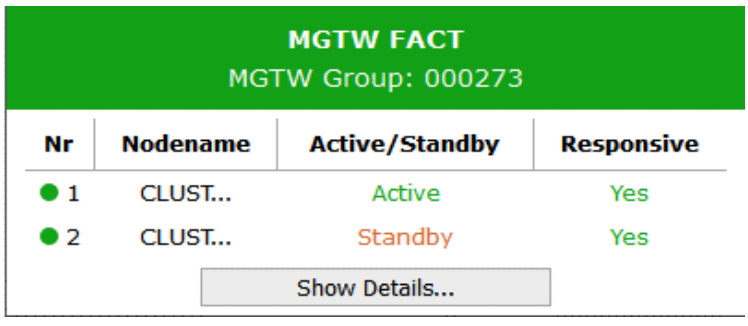



Figure 8 View Active-Standby Group Messaging Interface FACT

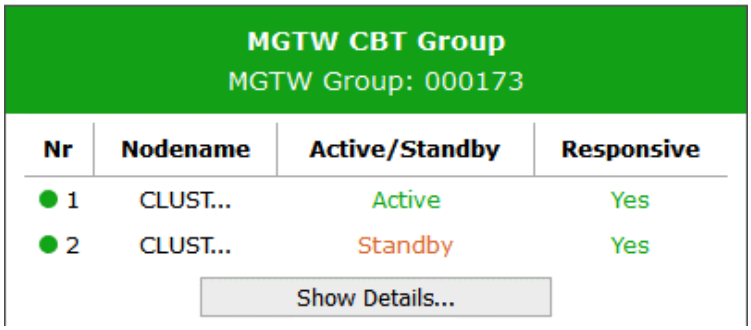



Figure 9 View Active-Standby Group Messaging Interface CBT

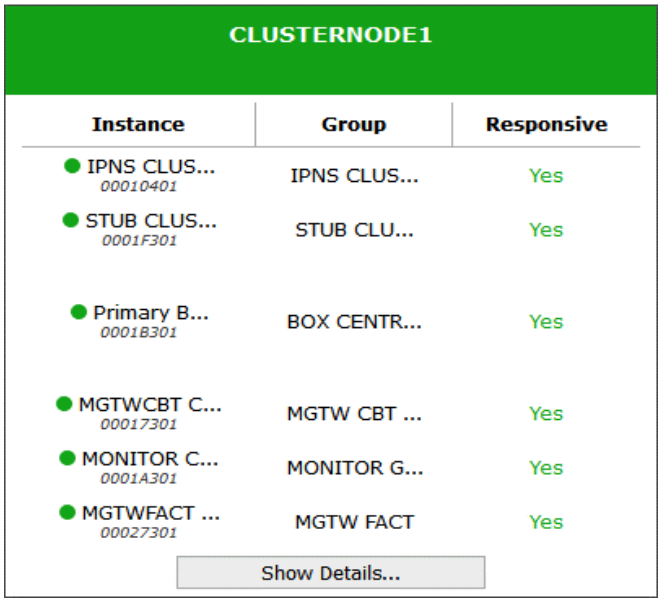



Figure 10 Node 1 View

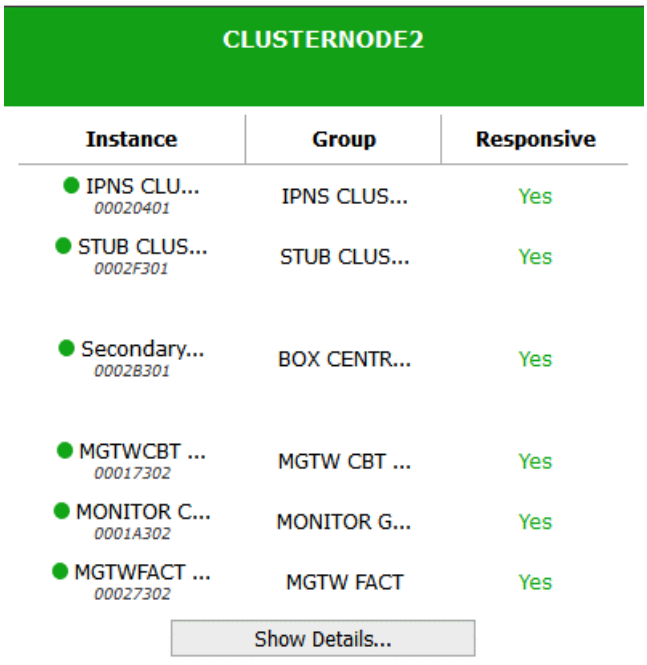
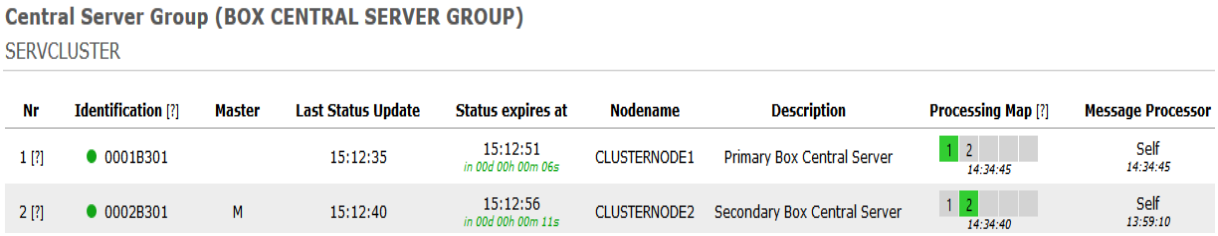



Figure 11 Node 2 View



**Central Server Group (BOX CENTRAL SERVER GROUP)**

SERVCLUSTER

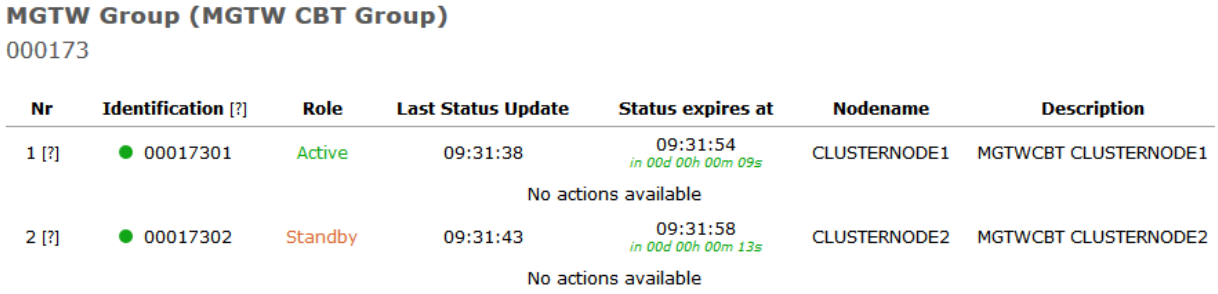
| Nr | Identification [?] | Master | Last Status Update | Status expires at | Nodename | Description | Processing Map [?] | Message Processor |
|---|---|---|---|---|---|---|---|---|
| 1 [?] | 0001B301 | | 15:12:35 | 15:12:51 *in 00d 00h 00m 06s* | CLUSTERNODE1 | Primary Box Central Server | 1 2 *14:34:45* | Self *14:34:45* |
| 2 [?] | 0002B301 | M | 15:12:40 | 15:12:56 *in 00d 00h 00m 11s* | CLUSTERNODE2 | Secondary Box Central Server | 1 2 *14:34:40* | Self *13:59:10* |

Figure 12 Detailed View of the Multi-Active-Group MPO_SERVER

## 1.5.3 Processing Map

Each Module Group view contains further details (Show Details…), which reflect the configuration of the arbitration. Each module role within the system can be viewed. The Box Central Server details also contain the 'Processing Map', which allows the user to identify the message processing module and to change the respective Box Server Module to take over the processing by means of resetting the 'Processing Map'.

### Example MGTW CBT Module Group



**MGTW Group (MGTW CBT Group)**

000173

| Nr | Identification [?] | Role | Last Status Update | Status expires at | Nodename | Description |
|---|---|---|---|---|---|---|
| 1 [?] | 00017301 | Active | 09:31:38 | 09:31:54 *in 00d 00h 00m 09s* | CLUSTERNODE1 | MGTWCBT CLUSTERNODE1 |
| | | | No actions available | | | |
| 2 [?] | 00017302 | Standby | 09:31:43 | 09:31:58 *in 00d 00h 00m 13s* | CLUSTERNODE2 | MGTWCBT CLUSTERNODE2 |
| | | | No actions available | | | |

Figure 13 Details of Module Group MGTW CBT

### Example Box Central Server Module Group

**Central Server Group (BOX CENTRAL SERVER GROUP)**

SERVCLUSTER

| Nr | Identification [?] | Master | Last Status Update | Status expires at | Nodename | Description | Processing Map [?] | Message Processor |
|---|---|---|---|---|---|---|---|---|
| 1 [?] | 0001B301 | | 09:52:35 | 09:52:51 *in 00d 00h 00m 13s* | CLUSTERNODE1 | Primary Box Central Server | 1 2 *09:48:55* | Self *09:50:18* |
| 2 [?] | 0002B301 | M | 09:52:30 | 09:52:46 *in 00d 00h 00m 08s* | CLUSTERNODE2 | Secondary Box Central Server | 1 2 *09:48:50* | Self *09:50:06* |

Figure 14 Details of Module Group Box Central Server

## 1.5.4 Reset Message Processor

The Box Central Server Processing Map indicates the Primary Server to process messages. The 'Reset' function is not usually required in a fully automated processing. It is used to manually initiated a takeover of Message Processing. Please refer to chapter 1.6.3 for details on a manual

takeover. The respective module can be highlighted using the ‘mouseover’ event. Select the module with mouse click and press reset.

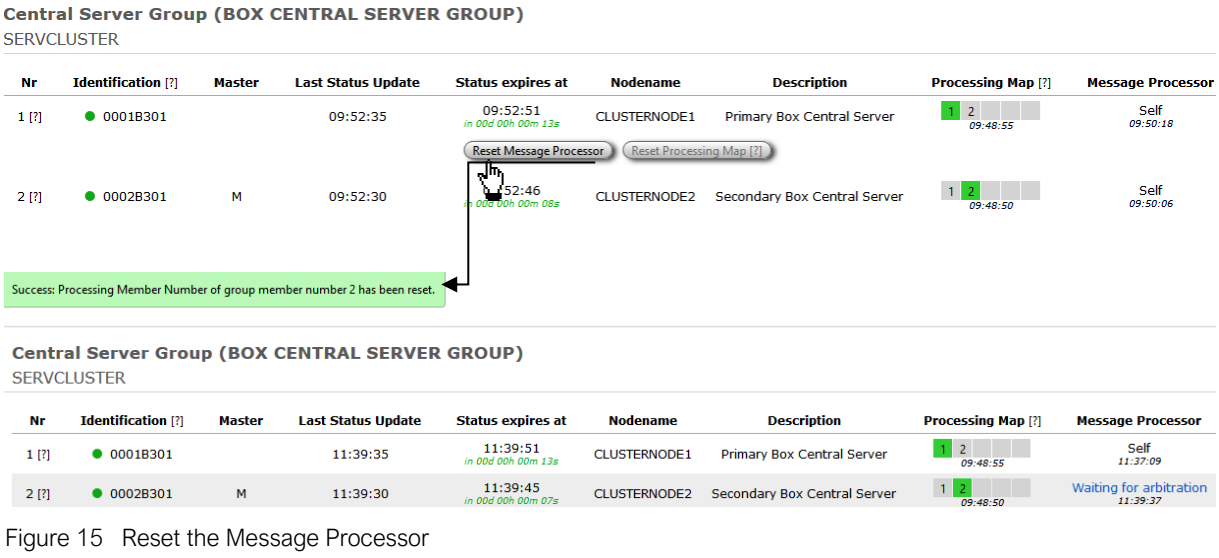



Figure 15 Reset the Message Processor

## 1.5.5 Reset Processing Map

In case of any arbitration hiccups, the Processing Map can be reactivated or changed by means of resetting it. Please refer to chapter 1.6.3 for details on an event of manual takeover.

IMPORTANT

The server has to be stopped completely before resetting the processing table.

To reset the Processing Map, all modules of Module Group Box Messaging Server have to be stopped.

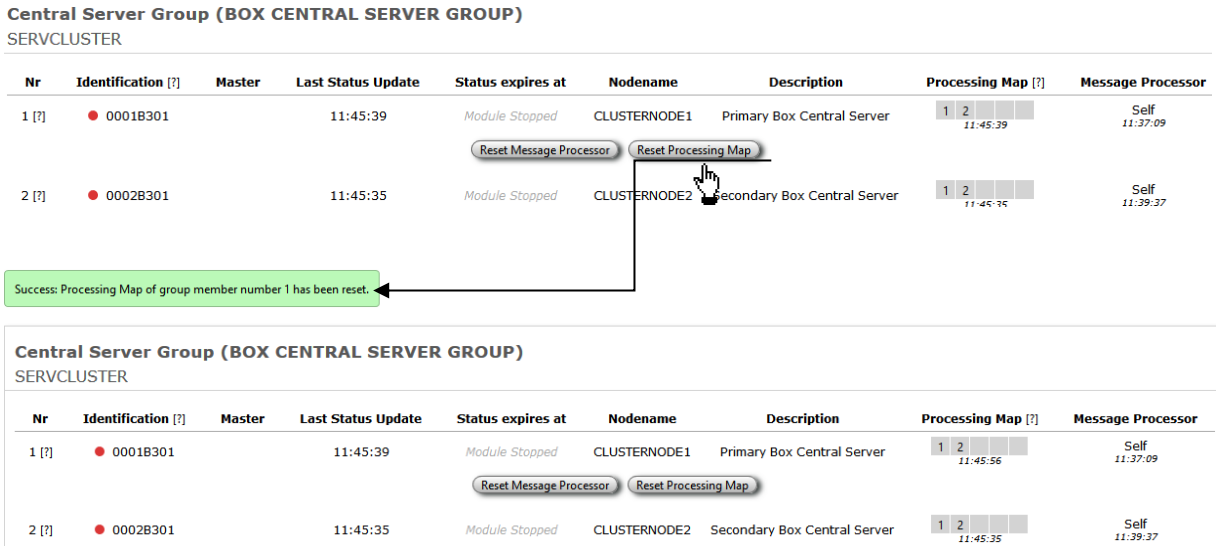



Figure 16 Reset the Processing Map

### 1.5.6 Overview Configuration

The central arbitration configuration is preferably stored in the shared database and maintained within the web client. It can also be maintained in the respective files. Options are implemented to serve the fine-tuning of the heartbeat and takeover, as well as the definition of Module Groups, Module Instances, Nodes and Mappings. The following graph gives an overview of the configuration of modules, sections, subsections and parameters.

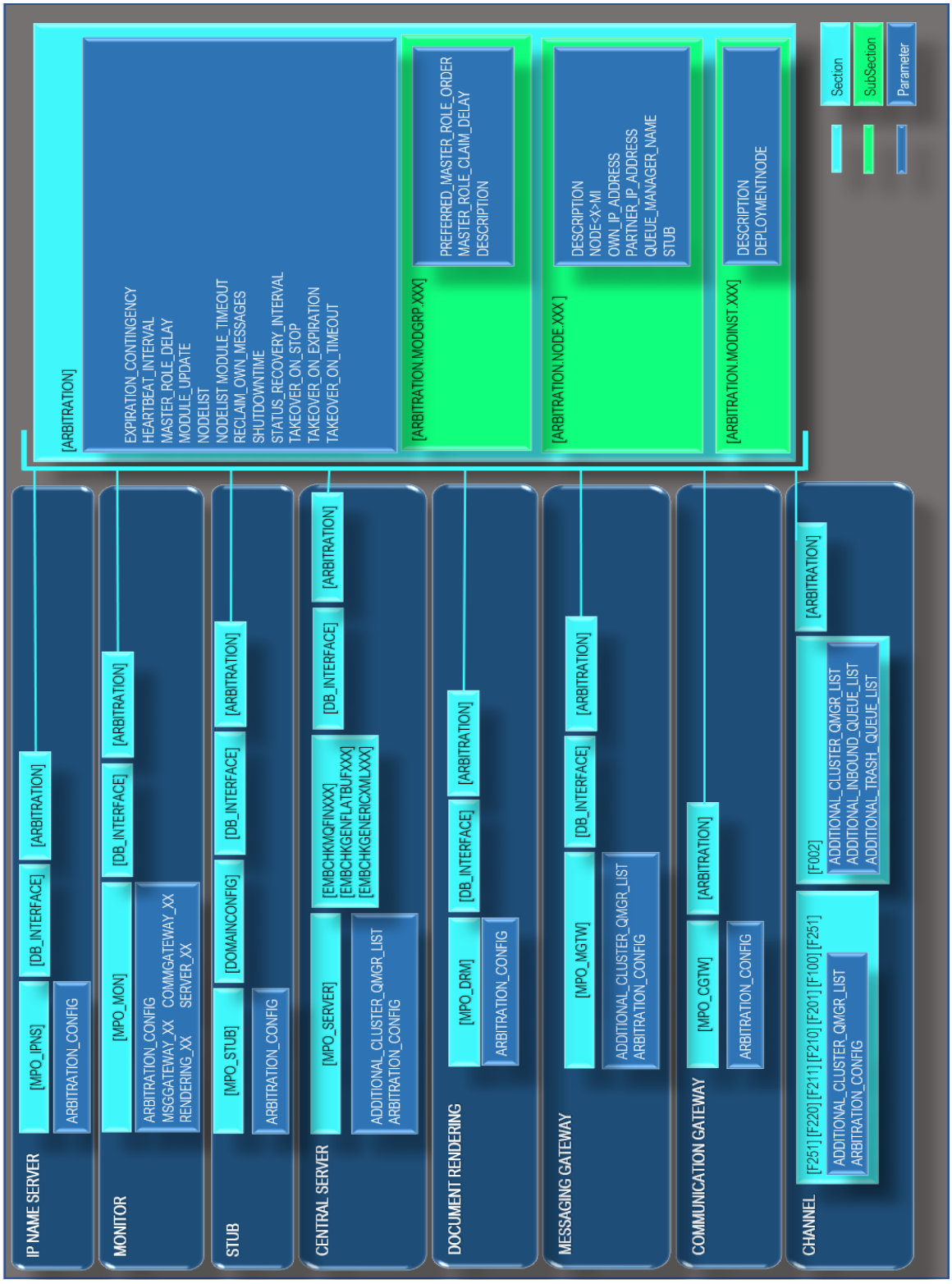



Figure 17 Graphical Overview: Hierarchical Configuration Structure

The same configuration applies for all Group Members, such as referencing in the arbitration configuration, e.g. ($ARB_CFG([NODE*$SELF*].IP_ADDRESS)).

**Please note, to enable module groups and arbitration a DB_INTERFACE section has to exist**

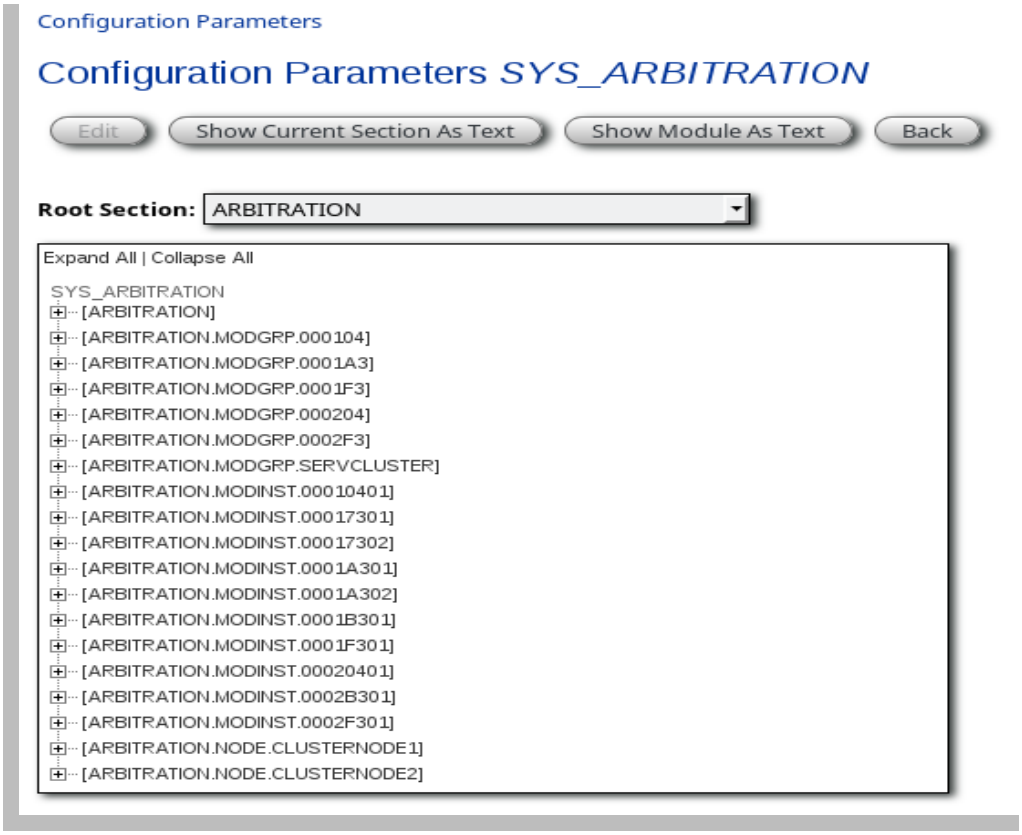



Figure 18 Web Client View on Configuration Parameters SYS_ARBITRATION

| Importer Type | Import Concurrency | Import Takeover | Remark |
|---|---|---|---|
| **MQ - Backoffice Interface** | Yes | N/A | |
| **File – Backoffice Interface** | Master only<br><br>Corporative Browsing | N/A<br><br>Yes | Corporative Browsing: ‚Hash' Import filename to Importer |
| **Database – Backoffice Interface** | Master only | N/A | |
| **MQ – Internal Central Server/MGTW (Response) Interface** | Yes | Yes | Fixed, programmed response-matching algorithm using MQ-Correlation |
| **MQ – Embargo (Response) Interface** | Yes | Yes | 3 different response matching algorithms possible (browsing, MQ-Correlation, Response-Queue) |

Table 4 Message Creation – Importer Concurrency

### 1.5.7 Upload of ARBITRATION Configuration in Database

The Arbitration section can be quite large depending on the environment and node number in use.

The recommended way to maintain this specific configuration is via the database. The tool mpoTransfer is designed to execute the upload not only of the server configuration, but also single sections, such as the section [ARBITRATION]. Please also refer to the respective mpoTransfer documentation for a concise description of the tool.

The following will suffice to upload the arbitration configuration and store it in the database:

```
./mpoTransfer.sh import_srvcfg -api configuration.properties -c SYS -u
Enterprise -p admin -rpl replace.rpl -sz security.zip  -cr server/ -i
arbitration.cfg -m SYS_ARBITRATION
```

IMPORTANT

Please be aware, that the value of -m (Module Name), here 'SYS_ARBITRATION' corresponds with the value of the parameter ARBITRATION_CONFIG, configured for each and every module. It is recommended to use a replacement token and configure the actual value in the file replace.rpl!

Example MPO_SERVER

```
[MPO_SERVER]
 ARBITRATION_CONFIG                $DB:SYS_ARBITRATION
```

### 1.5.8 The Active-Active Heartbeat

A heartbeat here refers to the Module Instances' regular status update within the arbitration table.

If planning a BOX Active-Active environment it is (still) recommended to use MQ-signalling when connecting Central Server modules to BOX-MI modules. It is now possible to configure database-signalling in a BOX Active-Active setup. To support this, further tables have been introduced.

If the configured Server Count is bigger than 1, then signals are read, and the server module id analysed. The server module ID is set for input messages and solicited output messages equally like MQ-signalling.

### 1.5.9 Cluster MQ Managers Configuration (including Embargo)

Queue Managers are configured to connect to each node either with a primary or secondary connection allowing for a clustered queue management.

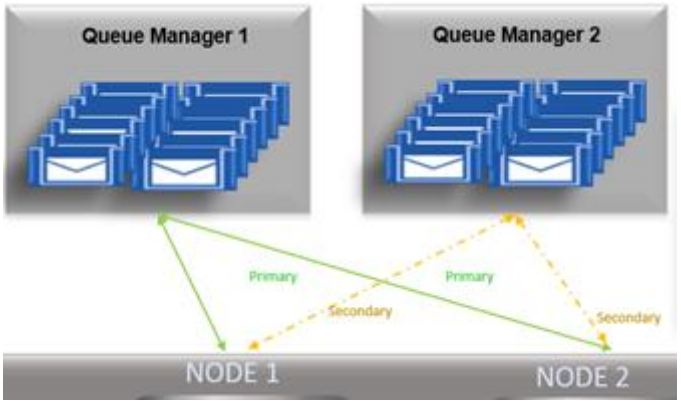



Figure 19 Queue Managers Connection to Node 1 and Node 2

#### 1.5.9.1 Connect to different cluster queue managers for message imports

- Use parameter **ADDITIONAL_CLUSTER_QMGR_LIST**

Use this list to specify additional (Cluster Queue) Managers from which response message shall be read from.

| Parameter ADDITIONAL_CLUSTER_QMGR_LIST configuration |
| --- |
| EMBARGO Configuration Sections:<br>[EMBCHKGENERICXMLXXX]<br>[EMBCHKGENFLATBUFXXX]<br>[EMBCHKMQFINXXX]<br><br>Use ADDITIONAL_CLUSTER_QMGR_LIST to specify additional (Cluster Queue) Managers from which response message shall be read from.<br><br>Make the number of retrieval tasks configured for the respective embargo check content processing plugin a multiple of the number of queue managers used (# of additional + 1 for local).<br>Parameters BROWSE_RESPONSE_QUEUE and RESPONSE_MATCHING (value MQALL) in this section are also used to specify embargo response matching when using MQ cluster. |
| Messaging Gateway Configuration Sections:<br>`[LCGZZZ.PEXA_SIGNAL]`<br><br>Use ADDITIONAL_CLUSTER_QMGR_LIST to specify additional (Cluster Queue) Managers which might be connected if MQ messages shall be sent by the MI module. Setting this parameter enables MQ cluster processing in MQ signalling on MI-module side. See also other parameters in this same section. |
| Exchange Adapter Configuration Sections:<br><br>`[F100]`<br>`[F201]`<br>`[F210]`<br>`[F211]`<br>`[F220]`<br>`[F251]`<br><br>Use ADDITIONAL_CLUSTER_QMGR_LIST to specify additional (Cluster Queue) Managers where MQ-signals shall be imported from. If using multiple inbound queue managers then the number of importers (parameter [LCGXXX].IMPORTER_COUNT) should be a multiple of (or same than) the number of queue managers listed here plus one (for the first queue manager). See also parameters SIGNAL_INBOUND_QUEUE and RESPONSE_INBOUND_QUEUE in this same section. |

Table 5 Parameter ADDITIONAL_CLUSTER_QMGR_LIST

### 1.5.9.2 Extended Configuration for F002 (eximf002)

The following parameters are used for a specific eximf002 configuration applied to several Input Queues.

- Connect to several inbound queues on same queue manager simultaneously using the same import configuration

  Use parameter ADDITIONAL_INBOUND_QUEUE_LIST and leave ADDITIONAL_CLUSTER_QMGR_LIST and ADDITIONAL_TRASH_QUEUE_LIST empty.

- Connect to several inbound queues on different queue managers simultaneously using the same import configuration

  Use parameter ADDITIONAL_INBOUND_QUEUE_LIST, ADDITIONAL_CLUSTER_QMGR_LIST, ADDITIONAL_TRASH_QUEUE_LIST.

## 1.6 Practical Example: Module Crash and Takeover

### 1.6.1 Initial Scenario

The initial status of the BOX Active-Active system presents it's with all modules being operational. The Processing Map shows, that every module performs processing of its own messages only and no Takeover is active. The Message Processor has created messages. There is currently no request for and no activation of the Takeover.

The graph below shows the Web Client representation of the Central Server Module Group.

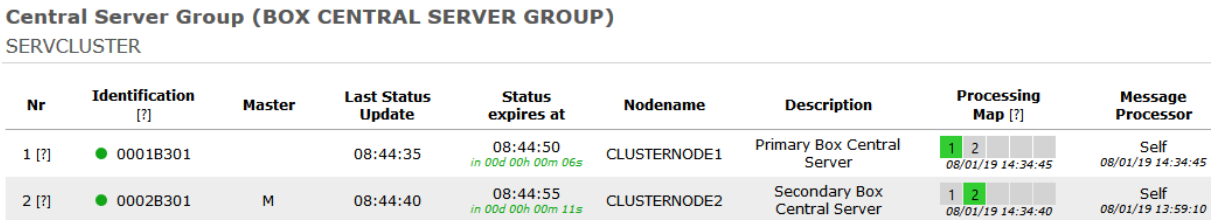



Figure 20 Web Client Representation: Module Group Details – Initial Scenario

### 1.6.2 Sudden Operative Change

The Primary Central Server holding the Master Role suddenly becomes unresponsive and is not available for processing anymore. The GUI indicates the failing module with color coding.

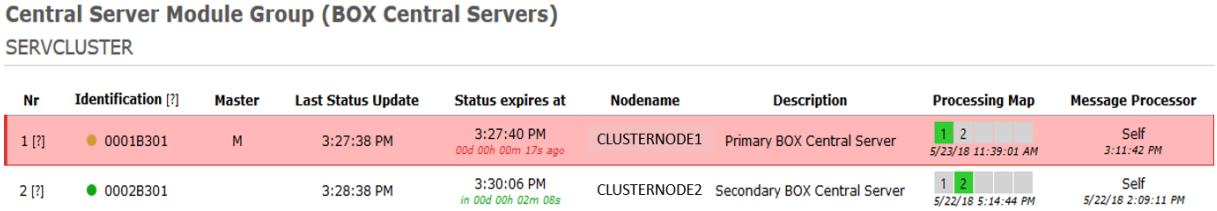



Figure 21 Web Client Representation: Module Group Details - Sudden Operative Change 1

The status is overall monitored of and by each individual module. The GUI also allows a more detailed view. The module row can be expanded to allow a view on the module actions.

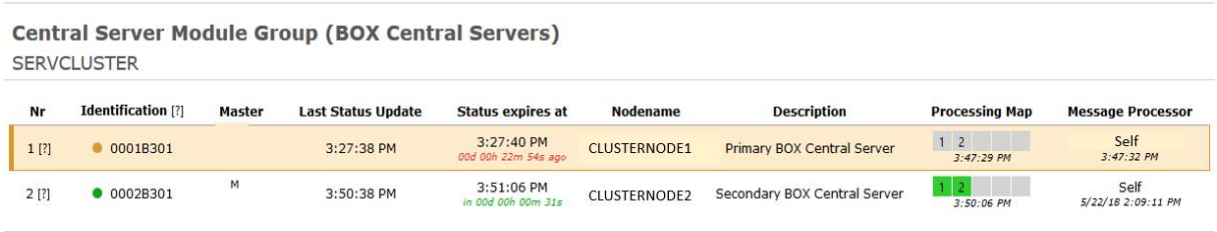



Figure 22 Web Client Representation: Module Group Details - Sudden Operative Change 2

### 1.6.3 Manual Takeover

1. Reset the Processing Map, which results in the withdrawal of the processing responsibility from the failed Module Member.
2. Reset the Message Processor, which allows stuck and pending messages to be available for other Group Members.

### 1.6.4 Automatic Takeover

If automatic Takeover on Expiration of status if configured, both actions of above step 1 and step 2 occur automatically.

### 1.6.5 Messaging Process Transferal

The Primary Central Server has been excluded from the Message Processing and respective messages are now made available for other Group Members.

**Central Server Module Group (BOX Central Servers)**
SERVCLUSTER

| Nr | Identification [?] | Master | Last Status Update | Status expires at | Nodename | Description | Processing Map | Message Processor |
|---|---|---|---|---|---|---|---|---|
| 1 [?] | 0001B301 | M | 3:27:38 PM | 3:27:40 PM<br>00d 00h 20m 15s ago | CLUSTERNODE1 | Primary BOX Central Server | 1 2<br>3:47:29 PM | Waiting for arbitration<br>3:47:32 PM |
| 2 [?] | 0002B301 | | 3:47:38 PM | 3:48:06 PM<br>in 00d 00h 00m 10s | CLUSTERNODE2 | Secondary BOX Central Server | 1 2<br>5/22/18 5:14:44 PM | Self<br>5/22/18 2:09:11 PM |

Figure 23 Web Client Representation: Module Group Details – Messaging Process Transferal

### 1.6.6 Completed Transferal of Message Processing

Additionally, the Secondary Central Server processes now messages of the Primary Central Server. As a consequence of this transferal, the Master Role now has changed to the Secondary Central Server, the Message Processing Map has been checked prior to Takeover and now is changed. After all, the Message Processor has now been changed.

**Central Server Group (BOX CENTRAL SERVER GROUP)**
SERVCLUSTER

| Nr | Identification [?] | Master | Last Status Update | Status expires at | Nodename | Description | Processing Map [?] | Message Processor |
|---|---|---|---|---|---|---|---|---|
| 1 [?] | 0001B301 | | 09:47:59 | Module Stopped | CLUSTERNODE1 | Primary Box Central Server | 1 2<br>09:47:59 | CLUSTERNODE2<br>09:48:00 |
| 2 [?] | 0002B301 | M | 09:48:00 | 09:48:16<br>in 00d 00h 00m 15s | CLUSTERNODE2 | Secondary Box Central Server | 1 2<br>09:48:00 | Self<br>08/01/19 13:59:10 |

Figure 24 Web Client Representation: Module Group Details – Message Transferal Completed

# 2 Practice

Considering the previous chapters, by now it might be clear, that great care must be taken during the implementation of Active-Active and the following chapters are meant to aid a perhaps in parts too detailed approach for the professional administrator, but it is our view, that considerations should at best include an overview of the envisaged system setup comprising a broad to very detailed view on the setup.

## 2.1 Outline the System Setup

The implementation of BOX running in an arbitrative mode must be preceded by careful planning of each individual part of this system. The following chapter will be based on an exemplary system of two servers running BOX and its clients and one server running the database. This system is a basic minimum advisable setup of running BOX.

All three individual parts must be maintained and updated to result in the same release of all BOX components, such as database, client and server with all their parts.

Since both BOX servers will represent the same setup including all configurations, it is recommended to maintain all configuration centrally in the database.

All configuration files can be imported into the database using the tool ‘mpoTransfer’, described in the manual box_transfer_v3r22.pdf.

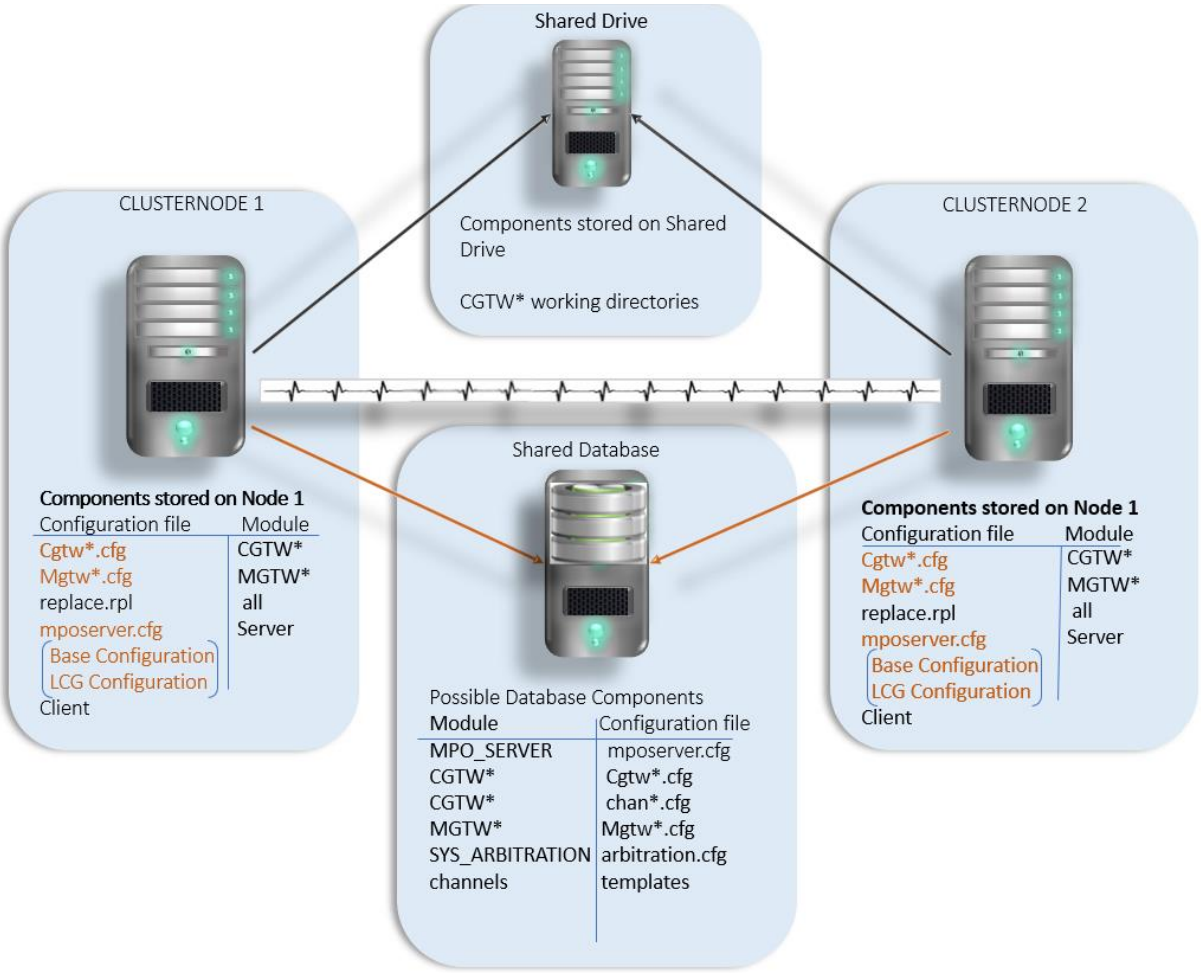



Figure 25 Components of a small arbitration system and their distribution

The above graph lines out a simple setup, whereby most configuration items are uploaded and stored in the database. The configuration requires a basic file of each module to reside in the

server's config directory. Within this file, it is determined to load further parameters from the database (marked in the above graph in brown). As of now, the Communication Gateway components should preferably be stored on a shared drive accessible to the Communication Gateways running on each Node.

### 2.1.1 Communication Name Changes

Communication names, used in Intra-MPO TCP/Pipes communication, now also include mmmmssii instead of mmmmss.

Additionally, a special instance number '0' is used for a specific purpose:

Modules may only use their 'real' instance number or only 0 or use both numbers.

**Stub-Module: 0 only**

Additionally, the active module instance initializes a communication interface using the special instance number 0.

**Exception: IPNS module, as this module uses a different protocol.**

## 2.2 Configuration Steps

### 2.2.1 Example Monitor configuration

```
[DOMAINCONFIG]
    SERVER_COUNT                    2
    SERVER_01                       T(1):1
    SERVER_02                       T(2):2

    RENDERING_COUNT                 0
    RENDERING_01

    MSGGATEWAY_COUNT                 2
    MSGGATEWAY_01                   $ARB_MODGRP:1,0x73;CBT
    MSGGATEWAY_02                   $ARB_MODGRP:2,0x73;FACT
```

Two servers configured

Arbitration for Two MGTW configured

Figure 26 Example mon.cfg Domain Configuration Section

### 2.2.2 Option /R

To start the active-standby modules in a specific active or standby mode, a new parameter '/R' has been introduced, which needs to be configured in the file …config/services.sh

To reflect the arbitration configuration, the primary module is configured with /R1 and the secondary with /R2.

**Node 1**

```
    ########            Monitor Module          ########
        dmon.sh)
        APP="${MPO_BASE}/bin/mpo_mon /I:1 /R1 /C:$CONFDIR/mon.cfg
/D:${LOGLEVEL} /S:${LOGSIZE} /L=$LOGDIR/mon.log "
        CONSOLE="${MPO_BASE}/bin/mpo_cout /I:1 /R1 /M:A3
/L:$LOGDIR/mpo_cout.log"
        ;;

    ########            MGTW     CBT        ########
        00017302.sh | mgtwcbt.sh)
```

```
        ulimit -c unlimited
        APP="${MPO_BASE}/bin/mpo_mgtw /I:1 /R1 /C:$CONFDIR/mgtwcbt.cfg
/D:${LOGLEVEL} /S:${LOGSIZE} /L=$LOGDIR/mgtwcbt.log "
        CONSOLE="${MPO_BASE}/bin/mpo_cout /I:1 /R1 /M:73
/L:$LOGDIR/mpo_cout.log"
        ;;

    ########            MGTW FACT             ########
        00027302.sh | mgtwfact.sh)
        ulimit -c unlimited
        APP="${MPO_BASE}/bin/mpo_mgtw /I:2 /R1 /C:$CONFDIR/mgtwfact.cfg
/D:${LOGLEVEL} /S:${LOGSIZE} /L=$LOGDIR/mgtwfact.log "
        CONSOLE="${MPO_BASE}/bin/mpo_cout /I:2 /R1 /M:73
/L:$LOGDIR/mpo_cout.log"
    ;;
```

**Node2**

```
    ########            Monitor Module          ########
        dmon.sh)
        APP="${MPO_BASE}/bin/mpo_mon /I:1 /R2 /C:$CONFDIR/mon.cfg
/D:${LOGLEVEL} /S:${LOGSIZE} /L=$LOGDIR/mon.log "
        CONSOLE="${MPO_BASE}/bin/mpo_cout /I:1 /R2 /M:A3
/L:$LOGDIR/mpo_cout.log"
        ;;

    ########            MGTW     CBT        ########
        00017302.sh | mgtwcbt.sh)
        ulimit -c unlimited
        APP="${MPO_BASE}/bin/mpo_mgtw /I:1 /R2 /C:$CONFDIR/mgtwcbt.cfg
/D:${LOGLEVEL} /S:${LOGSIZE} /L=$LOGDIR/mgtwcbt.log "
        CONSOLE="${MPO_BASE}/bin/mpo_cout /I:1 /R2 /M:73
/L:$LOGDIR/mpo_cout.log"
        ;;

    ########            MGTW FACT             ########
        00027302.sh | mgtwfact.sh)
        ulimit -c unlimited
        APP="${MPO_BASE}/bin/mpo_mgtw /I:2 /R2 /C:$CONFDIR/mgtwfact.cfg
/D:${LOGLEVEL} /S:${LOGSIZE} /L=$LOGDIR/mgtwfact.log "
        CONSOLE="${MPO_BASE}/bin/mpo_cout /I:2 /R2 /M:73
/L:$LOGDIR/mpo_cout.log"
    ;;
```

Figure 27 Option '/R' in Start Script (Example ..config/services.sh) to Specify InstanceNumber

### 2.2.3 Module Arbitration Parameter

Each Module running in an arbitrative mode has to be configured to do so.

Example MPO_SERVER

```
[MPO_SERVER]
 ARBITRATION_CONFIG                $DB:SYS_ARBITRATION
```

Example MGTW

```
[MPO_MGTW]
   ARBITRATION_CONFIG              $DB:SYS_ARBITRATION
```

### 2.2.4 Building Module Groups

Candidates for Module Groups are:

- IPNS
- MON
- STUB
- mposerver
- Messaging Gateways
- Communication Gateways (currently only supported with a shared drive)

Assumed are IPNS ID1 for Node1 and IPNS ID2 for Node 2:

| **Node 1** | | **Node2** | |
|---|---|---|---|
| **IPNS** | 000104 | IPNS | 000204 |
| **MONITOR** | 0001A3 | MONITOR | 0001A3 |
| **STUB** | 0001F3 | STUB | 0002F3 |
| **MGTW CBT** | 000173 | MGTW CBT | 000173 |
| **MGTW FACT** | 000273 | MGTW FACT | 000273 |
| **MPOSERVER** | SERVCLUSTER | MPOSERVER | SERVCLUSTER |
| | | | |
| | Runs independently from arbitration on both nodes | | |
| | Runs on both nodes active-standby | | |
| | Runs on both nodes active-active | | |

Table 6 Module IDs of Node1 and Node 2

### 2.2.5 Setting up Module Groups for arbitration

To set up the arbitration configuration, it is advised to create a configuration file, such as arbitration.cfg. In this file a correct arbitration can now be configured and, at a later stage, uploaded in the database.

Please note, as described earlier on, module ids are now extended to instance IDs to reflect the possibility to run up to five instances within one system.

The arbitration is hierarchically divided into

**Arbitration - Node - Module Group - Module Instance**

and within this division in module groups, in an active-standby mode and in an active-active mode.

## 2.2.6 Building the Arbitration Configuration File

### 2.2.6.1 Arbitration

```
[ARBITRATION]
  EXPIRATION_CONTINGENCY     5
  HEARTBEAT_INTERVAL         10
  MASTER_ROLE_DELAY
  MODULE_UPDATE
  NODELIST                   CLUSTERNODE1,CLUSTERNODE2
  RECLAIM_OWN_MESSAGES       YES
  SHUTDOWNTIME
  STATUS_RECOVERY_INTERVAL
  TAKEOVER_ON_EXPIRATION     NO
  TAKEOVER_ON_STOP           YES
  TAKEOVER_ON_TIMEOUT        NO
```

### 2.2.6.2 Nodes

```
[ARBITRATION.NODE.CLUSTERNODE1]
  DESCRIPTION          Primary Clusternode 1
  NODE1MI              NO
  NODE2MI              YES
  OWN_IPNS_PORT        1066
  OWN_IP_ADDRESS       192.168.56.231
  PARTNER_IP_ADDRESS   192.168.56.208
  QUEUE_MANAGER_NAME   $$R$CLNODE1QMGR$
  STUB                 TCP(1)


[ARBITRATION.NODE.CLUSTERNODE2]
  DESCRIPTION          Secondary Clusternode 2
  NODE1MI              YES
  NODE2MI              NO
  OWN_IPNS_PORT        1066
  OWN_IP_ADDRESS       192.168.56.208
  PARTNER_IP_ADDRESS   192.168.56.231
  QUEUE_MANAGER_NAME   $$R$CLNODE2QMGR$
  STUB                 TCP(2)
```

### 2.2.6.3 Module Groups

The following example combines all module groups necessary for the system.

Please note, for all module groups running in active-standby mode, the preferred order for master and slave (active and standby) has to be configured in 'PREFERRED_MASTER_ROLE_ORDER'.

```
[ARBITRATION.MODGRP.000104]
  DESCRIPTION    IPNS CLUSTERNODE1

[ARBITRATION.MODGRP.000204]
  DESCRIPTION    IPNS CLUSTERNODE2

[ARBITRATION.MODGRP.0001A3]
  DESCRIPTION     MONITOR GROUP
  PREFERRED_MASTER_ROLE_ORDER    CLUSTERNODE1, CLUSTERNODE2

[ARBITRATION.MODGRP.0001F3]
  DESCRIPTION     STUB  CLUSTERNODE1

[ARBITRATION.MODGRP.0002F3]
  DESCRIPTION     STUB CLUSTERNODE2

[ARBITRATION.MODGRP.SERVCLUSTER]
  DESCRIPTION     BOX CENTRAL SERVER GROUP
  PREFERRED_MASTER_ROLE_ORDER    CLUSTERNODE1, CLUSTERNODE2

[ARBITRATION.MODGRP.000173]
  DESCRIPTION     MGTW CBT Group
  PREFERRED_MASTER_ROLE_ORDER    CLUSTERNODE1, CLUSTERNODE2

[ARBITRATION.MODGRP.000273]
  DESCRIPTION     MGTW FACT
  PREFERRED_MASTER_ROLE_ORDER    CLUSTERNODE1, CLUSTERNODE2
```

| | |
|---|---|
| | Runs independently from arbitration on both nodes |
| | Runs on both nodes active-standby |
| | Runs on both nodes active-active |

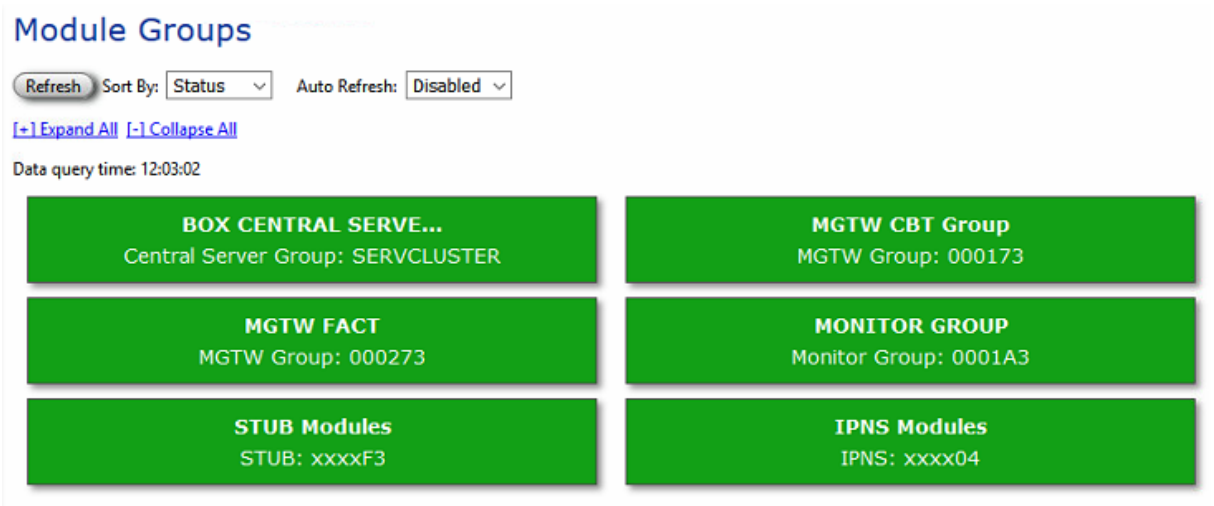



Figure Client View on Module Groups

#### 2.2.6.4 Module Instance

```
[ARBITRATION.MODINST.00010401]
  DEPLOYMENT_NODE   CLUSTERNODE1
  DESCRIPTION       IPNS CLUSTERNODE1

[ARBITRATION.MODINST.00020401]
  DEPLOYMENT_NODE   CLUSTERNODE2
  DESCRIPTION       IPNS  CLUSTERNODE2
```

```
[ARBITRATION.MODINST.0001A301]
  DEPLOYMENT_NODE   CLUSTERNODE1
  DESCRIPTION       MONITOR CLUSTERNODE1

[ARBITRATION.MODINST.0001A302]
  DEPLOYMENT_NODE   CLUSTERNODE2
  DESCRIPTION       MONITOR CLUSTERNODE2
```

```
[ARBITRATION.MODINST.0001F301]
  DEPLOYMENT_NODE   CLUSTERNODE1
  DESCRIPTION       STUB CLUSTERNODE1

[ARBITRATION.MODINST.0002F301]
  DEPLOYMENT_NODE   CLUSTERNODE2
  DESCRIPTION       STUB CLUSTERNODE2
```

```
[ARBITRATION.MODINST.0001B301]
  DEPLOYMENT_NODE   CLUSTERNODE1
  DESCRIPTION       Primary Box Central Server
```

```
[ARBITRATION.MODINST.0002B301]
  DEPLOYMENT_NODE   CLUSTERNODE2
  DESCRIPTION       Secondary Box Central Server
```

```
[ARBITRATION.MODINST.00017301]
  DEPLOYMENT_NODE   CLUSTERNODE1
  DESCRIPTION       MGTWCBT CLUSTERNODE1

[ARBITRATION.MODINST.00017302]
  DEPLOYMENT_NODE   CLUSTERNODE2
  DESCRIPTION       MGTWCBT  CLUSTERNODE2

[ARBITRATION.MODINST.00027301]
  DEPLOYMENT_NODE   CLUSTERNODE1
  DESCRIPTION       MGTWFACT CLUSTERNODE1

[ARBITRATION.MODINST.00027302]
  DEPLOYMENT_NODE   CLUSTERNODE2
  DESCRIPTION       MGTWFACT CLUSTERNODE2
```

| | |
|---|---|
| | Runs independently from arbitration on both nodes |
| | Runs on both nodes active-standby |
| | Runs on both nodes active-active |

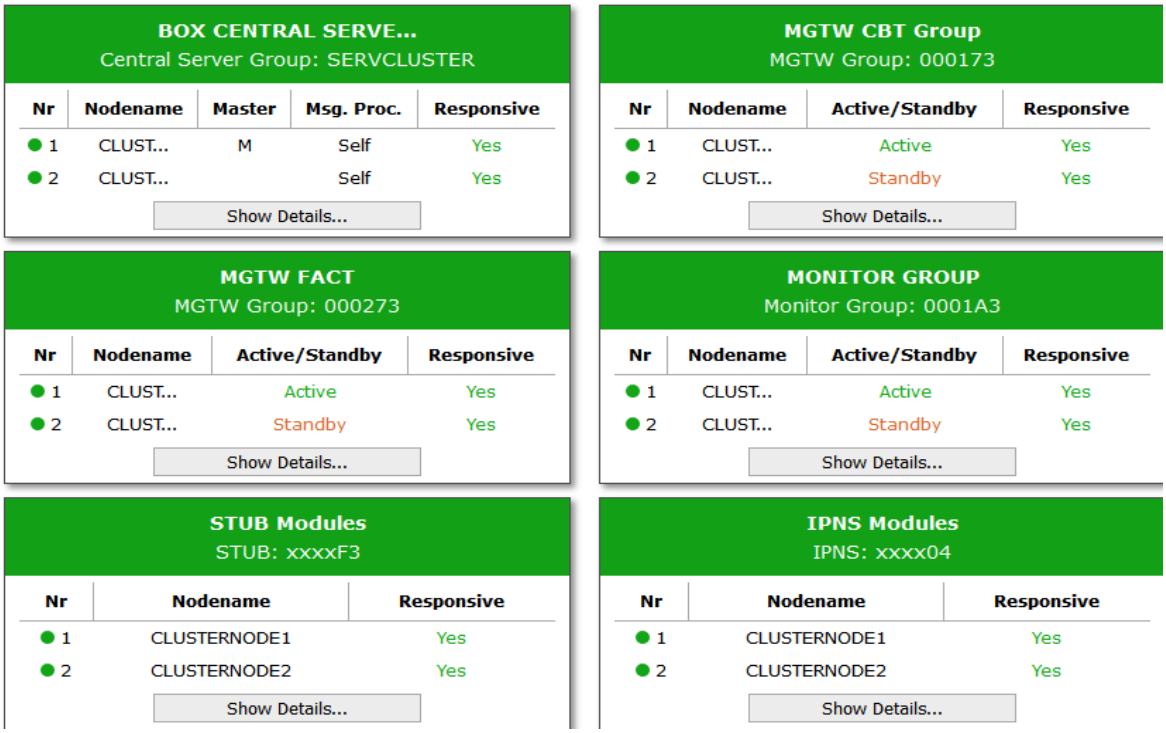



Figure 28 Client View on Module Instances

## 2.3 MQ-Backend Integration

The setup of Backoffice/GMA/MQ does not allow, that the MQ-IDs are used for the correlation with notifications.

In this case the correlation is possible using the following parameters:

| Module | Section | Parameter | Value |
|---|---|---|---|
| Exchange Adapter | F002 | RESPONSE_QUEUE | valid queue name or empty |
| If this parameter is specified than additional importer(s) are initialized reading this queue where (server-specific) responses are expected and read in a filtered fashion (see parameter RESPONSE_QUEUE_MATCHING ). This parameter has to be used with specific encoding plugins (e. g. SIAnet FEMS XS).<br><br>Parameter RESPONSE_QUEUE_MATCHING needs to be specified if a non-empty response queue is specified.<br><br>This parameter cannot be used together with ADDITIONAL_INBOUND_QUEUE_LIST in this section, i. e. only one of these parameters may be set. | | | |
| Exchange Adapter | F002 | SERVER_RESPONSE_MATCHING | Off |
| This parameter specifies how responses, read from the queue specified by parameter RESPONSE_QUEUE, are filtered (MsgId2CorrId). Set parameter only, if RESPONSE_QUEUE is not empty. | | | |

The server ID is coded into the message ID, which in turn is expected by the correlation ID.

Server are now able to filter the correlation ID.

# 3 Appendix

## 3.1 Configuration General Overview

### 3.1.1 Section ARBITRATION

The section arbitration applies to modules **Central Server, Communication Gateway, Document Rendering, IP Name Server, Monitor, Stub, Messaging Gateway**

Basic configuration parameters for module group definitions and arbitration.

To enable module groups and arbitration, a `DB_INTERFACE` section has to exist as well.

Parameter values in the sub-sections `(ARBITRATION.NODE.XXX, ARBITRATION.MODGRP.XXX and ARBITRATION.MODINST.XXX)` may be read in other (non-arbitration) configuration sections indirectly. In addition the modules own (self) node name, module group name or module instance name (iiiissrr) may be used in the module configuration via reference token from the arbitration configuration.

All but the very basic parameters can use this feature.

The sections where this cannot be used are:

- `[DB_INTERFACE]`
- `[SECURITY]`
- all sections with (optional) parameter `ARBITRATION_CONFIG`, e.g. `[MPO_SERVER], [MPO_MON]`

This indirection can be used by using the special "value filter function" $ARB_CFG in the parameter value.

#### 3.1.1.1 Syntax

```
$ARB_CFG( [<section>].<parameter> ) or $ARB_CFG($SELFNODE | $SELFMODGRP | $SELFMODINST)
```

For `<section>` and `<parameter>` any existing section/parameter combination in the arbitration configuration can be specified.

To address "special" sections in the arbitration configuration the following tokens can be used for <section>:

- `$SELFNODE` -> will be replaced by the full config section of the node the module instance belongs to
- `$SELFMODGRP` -> will be replaced by the full config section of the module group the module instance belongs to
- `$SELFMODINST` -> will be replaced by the full config section of the module instance

If necessary the value filter function can be used several times and be combined with constant text. The end result is the concatenation of all parts.

### 3.1.2 Section ARBITRATION_CONFIG

The section is configured for the IPNS (IP Name Server) in section [MPO_IPNS], STUB [MPO_STUB], section [MPO_MGTW] (Messaging Gateways), monitor section [MPO_MON], document rendering module section [MPO_DRM], communication gateways section [MPO_CGTW] and the Central Server [MPO_SERVER].

This parameter defines where the multi-server arbitration configuration (sections [ARBITRATION*]) shall be retrieved from. The following values can be used:
(empty): Use module base configuration file.

<pathname>: Read module configuration locally, using the supplied path.

$DB:<Parameter module name>: Read arbitration configuration from database.

For this specific module type (IPNS) this parameter is used to enable status advertisement to the central module table only. The module does not form an active standby-module group with any other stub module, but module status is visible in GUI via central module table.

### 3.1.3 Section ARBITRATION.NODE.<NODENAME>

This section is configured in the configuration of modules **Central Server, Communication Gateway, Document Rendering, IP Name Server, Monitor, Stub, Messaging Gateway**.

Configuration parameters specific for node hosting members of a module group. The node is specified by its name used in parameter [ARBITRATION].NODE_LIST.

### 3.1.4 Parameter Table

| Parameter | Value | Default | |
|---|---|---|---|
| Section: [ARBITRATION]<br><br>Modules:<br><br>Central Server,<br>Communication Gateway,<br>Document Rendering,<br>Monitor,<br>Messaging Gateway | | | |
| **STATUS_RECOVERY_INTERVAL** | **number of seconds, 0 allowed** | **60** | |
| This parameter might be used to avoid superfluous reassignment of resources after system start up or after recovery from database failure. This parameter can be used to define a time interval for which automatic takeover and master member arbitration for an individual member instance is disabled after this member has started or recovered from a database failure.<br><br>Example<br>STATUS_RECOVERY_INTERVAL 80 | | | |
| **EXPIRATION_CONTINGENCY** | **seconds** | **5** | |
| Additional time interval added to HEARTBEAT_INTERVAL to calculate module status expiration time. If the current time has passed the (status) expiration time a module is assumed to be unavailable and a master role arbitration may be started. In case of multi-active module groups messages assigned to the unavailable module may be reassigned and processing may be taken over by another group member (see TAKEOVER_ON_EXPIRATION and TAKEOVER_ON_STOP).<br><br>Example<br>EXPIRATION_CONTINGENCY 6 | | | |
| **SHUTDOWNTIME** | **seconds** | **120** | . |
| This parameter defines a time in seconds that is reserved for the arbiter thread to complete a shutdown<br><br>Example<br>SHUTDOWNTIME 180 | | | |

| Parameter | Value | Default |
|---|---|---|
| **HEARTBEAT_INTERVAL** | **seconds** | **30** |

The number of seconds between two consecutive status updates of a module in a module group.

Example
HEARTBEAT_INTERVAL 20

| | | |
|---|---|---|
| **RECLAIM_OWN_MESSAGES** | **YES, NO** | **NO** |

Switch to activate (automatic) takeover of own pending messages assigned to another module member in a module group.

Example
RECLAIM_OWN_MESSAGES NO

| | | |
|---|---|---|
| **MASTER_ROLE_DELAY (to be done yet)** | **seconds** | **0** |

Specify time allowance after DB-recovery to resync before any automatic reassignments occur.

Example
MASTER_ROLE_DELAY 5

| | | |
|---|---|---|
| **TAKEOVER_ON_STOP** | **YES, NO** | **NO** |

Switch to activate (automatic) takeover of pending messages assigned to a stopped module by another module member in a module group.

TODO: Comment on logic with stopped modules and restart.

Example
TAKEOVER_ON_STOP NO

| | | |
|---|---|---|
| **TAKEOVER_ON_EXPIRATION** | **YES, NO** | **NO** |

.

Switch to activate (automatic) takeover of pending messages assigned to an inactive module by another module member in a module group

Example
TAKEOVER_ON_EXPIRATION NO

| | | |
|---|---|---|
| **NODELIST** | **nodename1, nodename2, ..., nodename5** | **empty** |

Comma separated list of max. 5 deployment nodes available for grouped modules. An empty list will be handled as if the local hostname is configured here.

Important: Make sure that all module instances use the same node list with same names in same order!

Example
NODELIST BOXNODE1,BOXNODE2

Section: [ARBITRATION.NODE.<nodename>]

Modules:
Central Server,

| Parameter | Value | Default |
|---|---|---|
| Communication Gateway,<br>Document Rendering,<br>IP Name Server,<br>Monitor,<br>Messaging Gateway,<br>Stub | | |
| **STUB** | **<Protocol>(<Stub Module ID>)** | empty |

The Stub Module ID must be the Module ID of the Stub deployed on the node. Now the token $MODGRP may be used in parameters COMMGATEWAY_XX, MSGGATEWAY_XX and RENDERING_XX in Monitor configuration section [DOMAINCONFIG] to reference this information for all group members. For further details see description of parameters mentioned.

Example

```
[ARBITRATION.NODE.CLUSTERNODE1]
  DESCRIPTION           Primary Clusternode 1
  NODE1MI               NO
  NODE2MI               YES
  OWN_IPNS_PORT         1066
  OWN_IP_ADDRESS        192.168.56.231
  PARTNER_IP_ADDRESS    192.168.56.208
  QUEUE_MANAGER_NAME    $$R$CLNODE1QMGR$
  STUB                  TCP(1)
```

Section: [ARBITRATION.MODGRP.<groupname>]

Modules:
Central Server,
Communication Gateway,
Document Rendering,
Monitor,
Messaging Gateway

| **PREFERRED_MASTER_ROLE_ORDER** | **List of node names** | **empty** |
|---|---|---|

List of node names as used in [ARBITRATION].NODELIST. TBD

Example

```
[ARBITRATION.MODGRP.SERVCLUSTER]
  DESCRIPTION                   BOX CENTRAL SERVER GROUP
  PREFERRED_MASTER_ROLE_ORDER   CLUSTERNODE1, CLUSTERNODE2
```

| **MASTER_ROLE_CLAIM_DELAY** | **seconds** | **0** |
|---|---|---|

Delay in which the Master Role is taken over by the Slave

Example
Yet to be defined.

Section: [ARBITRATION.MODINST.MMMMSSII]

Modules:
Central Server,
Communication Gateway,
Document Rendering,
IP Name Server,

| Parameter | Value | Default |
|---|---|---|
| Monitor,<br>Messaging Gateway<br>Stub | | |
| **DEPLOYMENT_NODE** | **nodename as used in list** | **empty** |
| Specify the nodename (as used in parameter [ARBITRATION].NODE_LIST) where module instance is deployed (and operating) on. Hence the start-script, i.e. /R-runtime parameter, on this node has to use the correct (same) instance number as used in the section name.<br><br>This parameter is mandatory only if arbitration configuration is given.<br><br>Example<br>[ARBITRATION.MODINST.0001F301]<br>DEPLOYMENT_NODE CLUSTERNODE1<br>DESCRIPTION STUB CLUSTERNODE1<br><br>[ARBITRATION.MODINST.0002F301]<br>DEPLOYMENT_NODE CLUSTERNODE2<br>DESCRIPTION STUB CLUSTERNODE2 | | |
| | | |
| Section: [DOMAINCONFIG]<br><br>Module:<br>Monitor | | |
| **MSGGATEWAY_XX** | **$ARB_MODGRP<br>[,NOC]:<ModuleID>[,73]** | |
| In combination with active-active setup (arbitration) following applies starting with rel. 3.22.1:<br><br>Do not list each instance of a module (in a module group), use token $ARB_MODGRP instead of <Protocol>(<Stub Module ID>) in combination with parameter [ARBITRATION.NODE.<nodename>].STUB to declare and specify the active-standby module group related to this (logical) module. Otherwise single InstanceNumber 1 is implicitly assumed.<br><br>Example<br><br>[DOMAINCONFIG]<br>MSGGATEWAY_COUNT 2<br>MSGGATEWAY_01 $ARB_MODGRP:1,0x73;CBT<br>MSGGATEWAY_02 $ARB_MODGRP:2,0x73;FACT | | |
| **RENDERING_XX** | **$ARB_MODGRP<br>[,NOC]:<ModuleID>[,83]** | |
| In combination with active-active setup (arbitration) following applies starting with rel. 3.22.1:<br>Do not list each instance of a module (in a module group), use token $ARB_MODGRP instead of <Protocol>(<Stub Module ID>) in combination with parameter [ARBITRATION.NODE.<nodename>].STUB to declare and specify the active-standby module group related to this (logical) module. Otherwise single InstanceNumber 1 is implicitly assumed.<br><br>Example<br>[DOMAINCONFIG] | | |

| Parameter | Value | Default |
|---|---|---|

```
    RENDERING_COUNT                 2
    RENDERING_01                    $ARB_MODGRP:1,0x83
    RENDERING_02                    $ARB_MODGRP:2,0x83
```

| | | |
|---|---|---|
| **COMMGATEWAY_XX** | **$ARB_MODGRP[,NOC]:<ModuleID>[,93]** | |

In combination with active-active setup (arbitration) following applies starting with rel. 3.22.1:

Do not list each instance of a module (in a module group), use token $ARB_MODGRP instead of <Protocol>(<Stub Module ID>) in combination with parameter [ARBITRATION.NODE.<nodename>].STUB to declare and specify the active-standby module group related to this (logical) module. Otherwise single InstanceNumber 1 is implicitly assumed

Example

Please refer to MSGGATEWAY_XX.

| | | |
|---|---|---|
| **SERVER_XX** | **<Protocol>(<Stub Module ID>)[,NOC]:<ModuleID>[,<SocketID>]** | |

In combination with active-active setup (arbitration) following applies starting with rel. 3.22.1:

Every server module (with fixed InstanceNumber 1) has to be listed as separate server module. Token $MODGRP must not be used.

Example

```
[DOMAINCONFIG]
    SERVER_COUNT                    2
    SERVER_01                       T(1):1
    SERVER_02                       T(2):2
```

| Module | Section | Module | Section |
|---|---|---|---|
| Exchange Adapter | [F251] | Exchange Adapter | [F220] |
| Exchange Adapter | [F211] | Exchange Adapter | [F210] |
| Exchange Adapter | [F201] | Exchange Adapter | [F100] |
| Exchange Adapter | [F251] | Exchange Adapter | [F002] |

| | | |
|---|---|---|
| **ADDITIONAL_CLUSTER_QMGR_LIST** | **List of additional queue manager names (separate with ,)** | **empty** |

Please refer to chapter 1.5.9

```
Messaging Gateway    [LCGZZZ.PEXA_SIGNAL]
Central Server       [EMBCHKMQFINXXX]
```

| Parameter | Value | Default |
|---|---|---|
| Central Server [EMBCHKGENFLATBUFXXX]<br>Central Server [EMBCHKGENERICXMLXXX] | | |
| **ADDITIONAL_CLUSTER_QMGR_LIST** | **List of additional queue manager names (separate with ,)** | **empty** |
| Please refer to chapter 1.5.9 | | |
| | | |
| **Module**<br>Exchange Adapter [F002] | | |
| **ADDITIONAL_INBOUND_QUEUE_LIST** | **List of local queue names (separated by ',' )** | **Empty** |
| Please refer to chapter 1.5.9 | | |
| **ADDITIONAL_TRASH_QUEUE_LIST** | **List of trash queues associated to queue manager list (separated by ',' )** | **Empty** |
| Please refer to chapter 1.5.9 | | |

Table 7 List of Configuration Parameters

Intercope GmbH
Himmelstrasse 12-16,
22299 Hamburg,
Germany

+49 40 514 52 0
info@intercope.com

The Intercope home page can be found at
www.intercope.com