A MITEL
PRODUCT
GUIDE

MiVoice MX-ONE

System Database (Cassandra) - Description

Release 7.6
78/1551-ANF 901 14 Uen H

December 2023

Mitel
Powering connections

Notices

The information contained in this document is believed to be accurate in all respects but is not warranted by **Mitel Networks Corporation (MITEL®).** The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes. No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Mitel Networks Corporation.

Trademarks

The trademarks, service marks, logos and graphics (collectively "Trademarks") appearing on Mitel's Internet sites or in its publications are registered and unregistered trademarks of Mitel Networks Corporation (MNC), its affiliates, parents, or subsidiaries (collectively "Mitel") or others. Use of the Trademarks is prohibited without the express consent from Mitel. Please contact our legal department at legal@mitel.com for additional information. For a list of the worldwide Mitel Networks Corporation registered trademarks, please refer to the website:http://www.mitel.com/trademarks.

®,™ Trademark of Mitel Networks Corporation

Contents

Introduction 1

This chapter contains the following sections:

- General Cassandra Information
- Glossary and Acronyms

Scope

This document describes the general principle for how a distributed system database using eventual consistency is used in the MX-ONE Service Node.

The document also describes the most important files stored in the file system using the system database (Cassandra), and the basic principle of the tree structure used for the data stored in the system database.

1.1 General Cassandra Information

This document provides information only on those aspects of the use of the system database that are unique to the MX-ONE Service Node. For information about general use of the system database (Cassandra), you are recommended to read for example:

- Apache Cassandra web pages. http://cassandra.apache.org
- Datastax tutorial web pages about Cassandra. https://academy.datastax.com
- Cassandra: The Definitive Guide. Carpenter, Jeff, Hewitt, Eben. O'Reilly Media.

1.2 Glossary and Acronyms

ASP113

The product identity (abbreviated) of the MiVoice MX-ONE system. The full identity is ASP 113 01.

Cassandra Cluster

A collection of system database (Cassandra) data centers.

Cassandra Node, Node

A Cassandra instance running on a server.

Consistency Level

The minimum number of Cassandra nodes to reach to make sure a read or write request is consistent.

CQL

Cassandra Query Language, protocol used towards the system database.

CSV

Comma Separated Values, a data format used for example in .csv files, which store system database data.

Database

Here a Cassandra database concept which refers to the Apache Cassandra database.

Data Center

A geographical location of one or more Cassandra nodes.

Keyspace

A Cassandra concept for the control of the replication model(s) and the data in the database nodes

LIM

Line Interface Module, an MX-ONE Service Node entity, plus at least one media gateway. Can be co-located with a Cassandra node, but does not have to be.

Rack

A sub-division of servers within a Data Center, primarily to have separate power supply, or other redundancy.

Replication

The copying of data from a database node (replica) to another replica, making the relevant replicas as identical as possible.

Replication factor

The amount of copying of data from a database replica to other database replicas. A replication factor of 1 means that there is only one copy of each data in a Data Center, whereas a replication factor of 3 means three copies of the data are stored across the Data Center. Each Data Center has its own replication factor.

System Database

A Cassandra database concept which refers to the Apache Cassandra ™ database. Version 4.0.x is the latest when this document was last updated. Cassandra is open-source. Apache License 2 is used.

Table

A table in Cassandra is a distributed multi-dimensional map indexed by a key, used for the access of database data.

For a complete list of abbreviations and glossary, see the description for *Acronyms, Abbreviations, and Glossary*.

Overview 2

This chapter contains the following sections:

This section provides the architectural details of MiVoice MX-ONE system.

2.1 Architecture

The architectural model is to have at least one system database node accessible in every MiVoice MX-ONE Service Node serving the telephony applications of that Service Node (LIM). Therefore, read requests are directed to one or several system database nodes, either co-located with the Service Node, or located on a separate server as standalone system database node.

Any of the system database nodes can read/write, and all or some system database nodes can be replicas, depending on the keyspaces defined for the data which is read/written. Write operations need network access to at least one system database node. For full replication, the system needs access to all system database nodes.

In a telephony system with small amounts of configuration changes, one system database node can be located on one of the Service Node servers, as a system database node of that server. Other Service Nodes in the same MX-ONE system may have system database nodes, but they can also share a system database node with other Service Nodes.

In a telephony system with large amounts of configuration changes, the system database nodes may have to be located on a special external server (or servers). Every Service Node must have access to at least one system database node to be working, that is, to be able to read or write system database data.

If a system database node is out of service for a long time (for instance due to hardware failure), it is possible to manually reconfigure a new database node to replace the failing one (without loss of data). This is seen as a manual service measure. This process will not be automated.

The database functions are dependent on a reliable timeserver (NTP).

2.1.1 Scalability

The system database is designed to have read and write throughput both increase linearly as new machines are added, with the aim of no downtime or interruption to applications.

2.1.2 Firewall Considerations

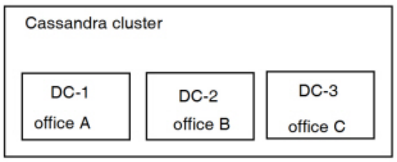
The keep alive timer is set to 7200 seconds (2 hours) for the Cassandra client sockets. Both for connections to port 9042 and 7001.

Normally, a Cassandra server has several connections to each of the other Cassandra servers in the database cluster. The sessions are open since start.

2.1.3 Cassandra Clusters

A Cassandra cluster holds one or more Data Centers. The MX-ONE system can support up to 40 data centers in a Cassandra cluster. The following figure shows a Cassandra cluster[1] with three data centers, named DC-1, DC-2, and DC-3.

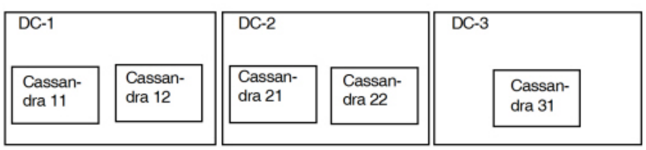
Figure 1: Cassandra Cluster with three Data Centers, named DC-1, DC-2 and DC-3



2.1.4 System Database Data Centers

A system database Data Center holds one or more system database nodes (servers). In the figure 2.2 DC-1 and DC-2 have two system database nodes, DC-3 has one system database node.

Figure 2: Cassandra nodes in each Data Center



Every system database node in a data center can have the same role. There is no single point of failure. Data is distributed across the data center, but there is no master node, as every node can service any request. This does not mean that every node store the same, identical data.

2.1.5 Data Replication

It is possible to specify to what nodes data should be replicated, that is, replication strategies are configurable. A common approach is to have a complete copy of all data in each Data Center. That is, a complete copy of all data in every geographical location. Internally in a specific Data Center, it is possible to specify on how many nodes a copy of a specific data should be stored. This is specified by the Replication Factor. With a Replication Factor of two, the data will be stored on two different system database nodes within the Data Center.

[1] The MX-ONE database structure, the way the Cassandra database (MX-ONE System Database) is setup, and how it is used by MX-ONE, has been changed in order to improve performance. As a result, new installations of MX-ONE 7.5 and later does now support configurations with up to 40 Data Centers (DC). In addition, MX-ONE 7.x systems upgraded to MX-ONE 7.5 and later now have support for configurations of up to 30 DC's. As the re-engineering of the system database (Cassandra) is so significant and wide-spread, the Cassandra changes can't be deployed through upgrading from MX-ONE 7.x to MX-ONE 7.5, or later. Therefore, upgraded MX-ONE systems will continue to have the original database structure and the original Cassandra setup, and therefore have the limitation of 30 DC's.

In a cluster like in figure 2, it is a good idea to have a Replication Factor of 2 for DC-1 and DC-2. With Replication Factor 2, the Data Center has internal redundancy of the data. DC-3 only has one system database node, so the Replication Factor will be one.

The total number of data copies that are replicated is referred to as the replication factor. A replication factor of 1 means that there is only one copy of each row in a Data Center, whereas a replication factor of 3 means three copies of the data are stored across the Data Center.

For ASP 113 data is always replicated to all database nodes in a Data Center. The replication factor for a Data Center must be equal to the number of Cassandra servers in the Data Center.

Data load sharing is not used in ASP 113.

2.1.6 Rack

A rack is the physical location of a server (or group of servers) within a Data Center. Servers are spread between racks to get redundancy if rack specific hardware fails. Examples of such hardware are power supplies, access switches and IP networks. The system database uses racks to spread replicas to different racks if they are available.

2.1.7 How Data is Written

When a write request is sent to one system database node, the write request is forwarded to all other system database nodes that hold a copy of that data table row. The system database only waits for a defined number of nodes to be ready with the write, before considering the write successful. This is defined in each write request as consistency level. If defined consistency level cannot be met, the write operation will fail.

If some nodes are not reachable, the system database has methods to make data consistent when the failed node has recovered. The system database stores a time-stamp for each write to know what the latest write is.

When a write is forwarded to another Data Center, each Data Center distributes the write requests to other nodes in the Data Center holding a copy of the data. This reduces the inter Data Center network traffic.

The system database tries to write to the node that has the shortest response time.

2.1.8 How Data is Read

When a read request is sent to one system database node, the request is forwarded to one node that holds a copy of the data. The system database also forwards digest requests to other Servers to verify that the data read is consistent with other replicas in the database. If the digest is consistent with the data read it is considered OK. The number of servers to get a correct answer from, before the read is considered successful, is defined in a read request consistency level. If defined consistency level cannot be met, the read operation will fail.

If the read data is not consistent with the digest, a repair action is started that will make this data table row consistent.

The system database tries to read from the node that has the shortest response time.

2.1.9 Tunable Consistence

Writes and reads offer a tunable level of consistency, all the way from *writes never fail* to *block for all replicas to be readable*, with the quorum level in the middle. The consistency level has to be adapted per function or application.

Consistency levels used by ASP 113:

For write: QUORUM.

For read: QUORUM, ONE.

Note:

The other consistency levels supported by Cassandra (such as ALL, EACH_QUORUM, LOCAL_QUORUM, TWO, THREE, ANY, SERIAL) are not used

2.2 System Database Deployment Examples in MX-ONE

The system database could when used in the ASP113 system be configured and deployed for example according to the following figures, either as a single Data Center, or as multiple Data Centers, and either co-located with the Service Node(s), or on separate server(s), standalone, and with one or multiple racks within a Data Center:

Figure 3: One Cassandra node co-located with a Service Node. One Data Center, and 1 rack. No redundancy.

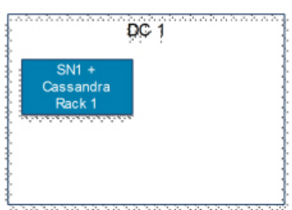


Figure 4: Four Cassandra nodes co-located and six Service Nodes in one ASP113 system. In two Data Centers (geographical locations), in 3 racks per DC. No server redundancy.

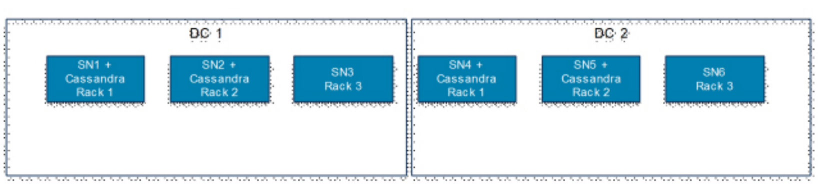


Figure 5: Cassandra, single site, single database, not co-located with SN. One Data Centre. No redundancy.

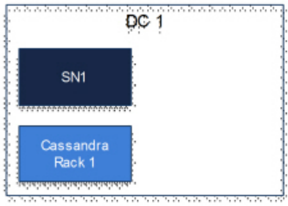


Figure 6: Cassandra, single site, multiple (2) databases, not co-located with SN. One Data Centre. Server redundancy (1+1) for SN1.

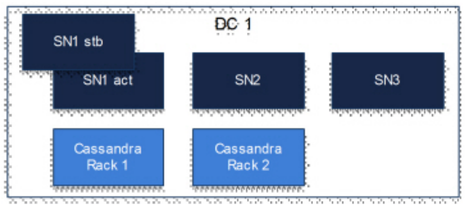


Figure 7: Cassandra with 2 DCs, 4 databases (2 per DC), standalone, i.e. not co-located with SN. 2 racks used per DC. A total of 7 Service Nodes, 6 active plus 1 standby which serves SN1 in DC1.

Figure 8: Cassandra in multi-site, co-located with the SNs, but only with one Cassandra node per DC. A total of 6 SNs and 6 DCs (geographical locations). No server redundancy.

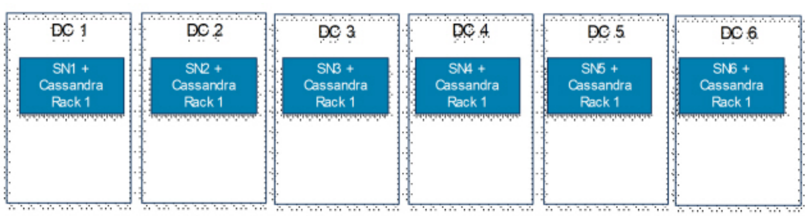


Figure 9: Cassandra in single site, the database is standalone (3 nodes). 1 Data Center with 3 racks, and a system with 6 Service Nodes. No server redundancy.

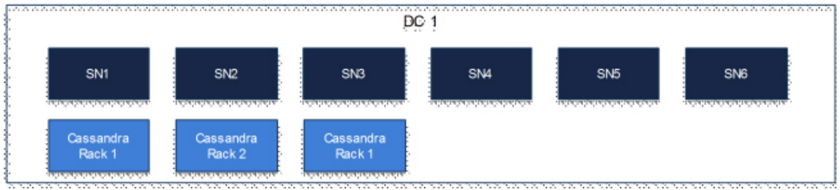


Figure 10: Cassandra with 2 racks each, where DC2 is remote, and server redundancy is used in both DCs, that is, SN1 and SN4 have standby servers. There is an extra Cassandra node in rack 2 of each DC, which could be skipped.

There are of course many more possible configurations, with up to 40 DCs, with co-located or stand-alone Cassandra nodes, with single or multiple racks, and with or without standby-SN.

2.3 System Database Data Model Details

System Database Principles

This section briefly describes the system database data model.

See References on page 16 for further details.

Decentralized Database

Every node in the cluster can have the same role. There is no single point of failure. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.

Every Service Node must have access to one or several system database nodes. The system database nodes do not have a 1-to-1 relation to the MX-ONE Service Nodes. Instead the number of system database nodes should be kept as low as possible, for example, in a system with 10 Service Nodes, no more than 3 or 4 system database nodes should be used. For redundancy, a minimum of 2 system database nodes is recommended.

System Database Data Replication

The replication protocol of Cassandra is used. In this way all the necessary information is available locally on every system database node (replica) which is defined to have the particular data.

The concept of a Cassandra database is that it is distributed, built to work decentralized, scalable and with high availability. Replication is done without a master-slave relationship, in a “ring” architecture per Data Center, but not necessarily so that all nodes store the same data, controlled by the used keyspaces.

Replication strategies are configurable. The system database is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of the system database’s distributed architecture are specifically tailored for multiple-data center deployment, for redundancy, for failover and disaster recovery.

A Cassandra cluster can have one or more keyspaces, which are analogous to Microsoft SQL Server and MySQL databases or Oracle schema. Replication is configured at the keyspace level, allowing different keyspaces to have different replication models.

The system database is able to replicate data to multiple nodes in a cluster, which helps ensure reliability, continuous availability, and fast management operations. The total number of data copies that are replicated is referred to as the replication factor. A replication factor of 1 means that there is only one copy of each row in a Data Center, whereas a replication factor of 3 means three copies of the data are stored across the Data Center.

Once a keyspace and its replication have been created, the system database automatically maintains that replication even when nodes are removed, or added.

Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed Cassandra nodes can be replaced with no downtime.

Cassandra Query Language, CQL, is used when accessing the database.

Files 3

This chapter contains the following sections:

The files related to the use of the system database are data files, system database configuration files, and the start script.

3.1 Data Files for the System Database

Data files related to the system database are located in the directories:

`/var/opt/cassandra/data`

`/var/opt/cassandra/commit`

`/var/log/cassandra/`

3.2 System Database Configuration Files

The system database configuration files are:

`/usr/share/cassandra/conf/cassandra.yaml`

`/usr/share/cassandra/conf/cassandra-env.sh`

`/usr/share/cassandra/conf/cassandra-rackdc.properties`

`/usr/share/cassandra/conf/cassandra-topology.properties`

`/usr/share/cassandra/conf/jvm.options`

Handling of mxone_data and mxone_global data 4

There are 2 locations where the data is stored in the MX-ONE Service Node. They are: mxone_data (profile data) and mxone_global (currently active feature).

- mxone_data the profile data, csv-files data, (managed with command `-i/c/e/p`) is what the administrator configures using commands and this data is included in the data backup of the system. Example, *diversion -i -d 10410 --div-destination-number 10701*.
- mxone_global currently active feature (printed with command `-p` and specific command). Data is set by user through procedures or CSTA-requests. This data is not included in the data backup of the system. Example, *diversion_info -p -d 10410*.

The mxone_global data holds information about currently active feature, and it means that the active feature data will not be lost at system actions as system upgrade, LIM/system reload, power failure, and switch-over to standby server. Once the system is back in service, the active feature will be the same as before the system action. Active feature data are lost only at re-installation of the system or at repair of a server including all Cassandra data base in the system (for example, 1 lim system with co-located database).

Backup Strategy 5

This chapter contains the following sections:

The backup strategy for the system database-based data is integrated with the backup strategy for old style reload data in the MX-ONE Service Node. There is no separate backup or restore done for the system database data. The `data_backup` and `data_restore` commands affect both reload data and system database data.

5.1 The data_backup Command

When the `data_backup` command is issued, the Cassandra data is dumped per database table to a number of CSV files, using the CQL protocol. The CSV (.csv) files are stored locally in each LIM, also if the LIM has no local Cassandra node, just as the .D files for reload data.

When a data reload is executed (either as a result of the `data_restore` command, or as a system measure), the data in the master LIM (server) is rolled back to match the CSV files. This is done by running a rollback command behind the scenes. Then it does only the necessary changes to make the data in the Cassandra node(s) match the CSV files.

A Cassandra node can read data from the CSV files. The CSV files are created on every LIM, in case a Cassandra entity has to be moved due to hardware failure.

5.2 Safety Backup

The data dumped to the local hard disk by the data backup function shall be backed up to some external storage as a safety backup. This data will be in CSV format.

This procedure is not described here as it is not Cassandra-specific.

The `config_mirror` script also does the same, plus more.

Starting and Stopping System Database

6

To start, restart or stop system database nodes, use the standard SLES/Linux systemd commands with the function `mxone_db.service`.

The `mxone_maintenance` script contains functions for add and remove of system database nodes.

Error and Management Handling on the System Database 7

The system database has its own dedicated management tools, such as nodetool. This standard Linux systemd tool has functions for compaction, cleanup, status check, failure detection, flush and repair.

The used Cassandra configuration has certain limitations, which mean it is not allowed to connect the nodetool (remotely) from other servers than the one where a Cassandra node is running. See the *nodetool online help* in a server with Cassandra.

The *mxone_maintenance* script contains functions for Adding a database node to server, and Removing a database node from server.

Alarms 8

The system database operation is supervised and alarms are generated for the following error conditions:

- Fault Code (1:53) Cannot read from system database.
- Fault Code (1:55) System database out of order.
- Fault Code (1:56) NTP state is not correct.
- Fault Code (1:57) System database schema version mismatch between hosts.

References 9

The following reference documents concerning the system database are relevant:

Table 1: Reference documents concerning the system database

1.	Cassandra: The Definitive Guide. Carpenter, Jeff, Hewitt, Eben. O'Reilly Media.
2.	Datastax tutorial web pages about Cassandra 4.x. https://academy.datastax.com
3.	Apache Cassandra web page. http://cassandra.apache.org

See also the document *Fault location*, regarding alarms.