|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Table of Contents

[1 Introduction 6](#_Toc55221628)

[1.1 Purpose 6](#_Toc55221629)

[1.2 Scope 6](#_Toc55221630)

[1.3 Definitions and Acronyms 6](#_Toc55221631)

[1.4 References 8](#_Toc55221632)

[2 System High Level Configuration 9](#_Toc55221633)

[2.1 Architecture Styles 10](#_Toc55221634)

[2.1.1 Architecture – Deployment 10](#_Toc55221635)

[2.1.1.1 Application Servers 12](#_Toc55221636)

[2.1.1.2 Workstations 13](#_Toc55221637)

[2.1.1.3 Networking 13](#_Toc55221638)

[2.1.1.4 External Interfaces 13](#_Toc55221639)

[2.1.2 Architecture – Inter-process Communication 13](#_Toc55221640)

[2.1.2.1 EBI Screen Proprietary MoM - UISR 15](#_Toc55221641)

[2.1.2.2 Third-party MoM - Artemis 17](#_Toc55221642)

[2.1.2.3 CTC – TMS IPC interoperability 19](#_Toc55221643)

[2.1.3 Architecture - Structure 20](#_Toc55221644)

[2.2 Software Control Implementation 21](#_Toc55221645)

[2.3 Application Control 21](#_Toc55221646)

[2.3.1 CTC Application Server Control 21](#_Toc55221647)

[2.3.2 TMS Application Server Control 23](#_Toc55221648)

[2.3.2.1 TMS Cluster configuration 24](#_Toc55221649)

[2.3.3 Workstation Control 25](#_Toc55221650)

[2.3.4 Backup Control Centre Control 26](#_Toc55221651)

[3 Architecture Design 27](#_Toc55221652)

[3.1 System Context 27](#_Toc55221653)

[3.1.1 ATS Operator 27](#_Toc55221654)

[3.1.2 Wayside Signalling 27](#_Toc55221655)

[3.1.2.1 EBI Lock 950 R4 27](#_Toc55221656)

[3.1.3 Wayside ATO 28](#_Toc55221657)

[3.1.4 Wayside TCC 28](#_Toc55221658)

[3.1.5 Video Walls 28](#_Toc55221659)

[3.1.6 Master Clock 28](#_Toc55221660)

[3.1.7 Project-specific Internal Systems 28](#_Toc55221661)

[3.1.7.1 CITYFLO 150 Onboard Subsystem 28](#_Toc55221662)

[3.1.7.2 EBI Lock 400 28](#_Toc55221663)

[3.1.8 Project-specific External Systems 29](#_Toc55221664)

[3.2 System Decomposition 29](#_Toc55221665)

[3.3 Subsystem Division and Referral 30](#_Toc55221666)

[3.3.1 Ebibase 30](#_Toc55221667)

[3.3.2 Ebicore 31](#_Toc55221668)

[3.3.3 Interfaces 33](#_Toc55221669)

[3.3.3.1 Wayside Interfaces 33](#_Toc55221670)

[3.3.3.2 Onboard Interfaces 34](#_Toc55221671)

[3.3.3.3 Combined Onboard / wayside Interfaces 35](#_Toc55221672)

[3.3.4 Tools and Simulators 36](#_Toc55221673)

[3.3.5 Traffic Scheduling System 39](#_Toc55221674)

[3.3.5.1 Subsystems in TMS Server 39](#_Toc55221675)

[3.3.5.2 Offline Simulation 40](#_Toc55221676)

[3.3.6 ATS User Interface Subsystem (ATSUI) 41](#_Toc55221677)

[3.3.6.1 Context 41](#_Toc55221678)

[3.3.6.2 Decomposition 41](#_Toc55221679)

[3.3.6.3 Configuration 42](#_Toc55221680)

[3.3.6.4 Interfaces 43](#_Toc55221681)

[3.3.6.4.1 CTC User Interface (CTCUI) 43](#_Toc55221682)

[3.3.6.4.2 CTCUI Architecture 44](#_Toc55221683)

[3.3.6.5 Event/Alarm Management User Interface (AMUI) Applications 44](#_Toc55221684)

[3.3.6.5.1 AMUI Architecture 45](#_Toc55221685)

[3.3.6.6 Traffic Scheduler User Interface 45](#_Toc55221686)

[3.4 Implementation Rules 45](#_Toc55221687)

[3.4.1 Operating System 46](#_Toc55221688)

[3.4.2 Programming Languages 46](#_Toc55221689)

[3.4.3 Internationalization 46](#_Toc55221690)

[3.4.4 Association Implementation 46](#_Toc55221691)

[4 Architecture Components 47](#_Toc55221692)

[5 Interfaces 56](#_Toc55221693)

[5.1 External Interfaces 56](#_Toc55221694)

[5.2 Internal Interfaces 56](#_Toc55221695)

[6 ARCHITECTURE MANAGEMENT 57](#_Toc55221696)

[6.1 Memory Management 57](#_Toc55221697)

[6.2 Handling of Global Resources 57](#_Toc55221698)

[6.3 Identification of Concurrence 57](#_Toc55221699)

[6.4 Selection of Architectural Framework 57](#_Toc55221700)

[6.5 Handling of Boundary Conditions 57](#_Toc55221701)

[6.5.1 Initialisation 57](#_Toc55221702)

[6.5.2 Termination 58](#_Toc55221703)

[7 Architecture Safety 59](#_Toc55221704)

[7.1 Safety Strategy 59](#_Toc55221705)

[7.2 Safety Techniques 59](#_Toc55221706)

[7.3 Failure 59](#_Toc55221707)

[8 Architecture Performance 61](#_Toc55221708)

[8.1 Scheduling Capacity 61](#_Toc55221709)

[8.2 Estimating Hardware Resource Requirements 61](#_Toc55221710)

[8.3 Hardware-Software Trade-offs 61](#_Toc55221711)

[8.4 Throughput 61](#_Toc55221712)

[8.5 Storage 61](#_Toc55221713)

[8.6 Trade-Off Priorities 61](#_Toc55221714)

[9 Architecture Reliability 62](#_Toc55221715)

[10 Architecture Maintainability 63](#_Toc55221716)

[10.1 Test Features 63](#_Toc55221717)

[10.2 Diagnostics and Fault Reporting 63](#_Toc55221718)

Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date (yyyy-mm-dd) | Document Producer | Description of Changes |
| 1.0A | 2016-08-30 | Jari Lintunen | Document item created |
| 1.0A | 2017-08-10 | A Härtsiä | 4.0 / BHPB update |
| 1.0A | 2018-03-13 | A Härtsiä | Mail and Note added, BHPB stage 1 specific items removed |
| 1.0A | 2018-03-29 | A Härtsiä | BHPB specific applications (Train Configuration server and user interface, Genisys interface) removed from chapter 4, changed mrt-purple-line.dll to tms-e2k-converter.dll (TMS specific converter). |
| 1.0A | 2018-04-11 | A Härtsiä | Updated after review |
| 1.0A | 2020-10-22 | M Forss | Updated, mainly MoM related issues. |
| 1.0A | 2020-11-02 | M Forss | Updated after review |

# Introduction

## Purpose

This document specifies the software architecture of the EBI Screen 2000 ATS product. This encompasses how the software is structured and scheduled, and description of internal and external interfaces.

The main function of EBI Screen 2000 is to support the operational personnel with operation-relevant information, processing, automatism and command capability related to standard activities during normal and abnormal operational situations.

## Scope

This document describes the software architecture features and components of EBI Screen 2000.

More detailed information is presented in subsystem specific documents.

## Definitions and Acronyms

|  |  |
| --- | --- |
| Definitions: |  |
|  |  |
| Acronyms: |  |
| Artemis | Apache ActiveMQ Artemis, an example of Message Oriented Middleware |
| ATC | Automatic Train Control |
| ATR | Automatic Traffic Regulation |
| ATS | Automatic Train Supervision |
| BATS | Central ATS in Backup Control Centre (BCC) |
| BCC | Backup Control Centre |
| CATS | Central ATS in Operation Control Centre (OCC) |
| CTC | Centralized Traffic Control |
| DLL | Dynamic Link Library |
| DTS | Data Transmission Network |
| ICD | Interface Control Document |
| HTTP | Hypertext Transfer Protocol |
| IPC | Inter-process Communication |
| JMS | Java Messaging Services |
| Jakarta EE | A set of specifications that enables the worldwide community of Java developers to work on Java enterprise applications. |
| LAN / WAN | Local Area Network / Wide Area Network |
| MDB | Message Driven Bean. |
| MDC | Maintenance and Diagnostic Centre. |
| MD5 | The MD5 Message-Digest Algorithm, cryptographic hash function that produces a 128-bit (16-byte) hash value. Used to check data integrity. |
| MoM | Message Oriented Middleware |
| N/A | Not Applicable |
| OCC | Operation Control Centre |
| PAS | Passenger Information System |
| Payara | An enterprise application server that implements Jakarta EE specification. |
| PI | Process Interface |
| RCS | Rail Control Solutions |
| REST | Representational State Transfer |
| RIPEMD4-160 | 160-bit message digest algorithm |
| RSO | Network position based on Region, Segment, Offset |
| SCADA | Supervisory Control Automatic Data Acquisition |
| SIL | Safety Integrity Level |
| SMT | Site Management Tool |
| SOA | Service-Oriented Architecture; architecture based upon service-oriented applications. |
| SOAP | Single Object Access Protocol: message-encoding protocol based on XML technologies, defining an envelope for web service communication. |
| STOMP | Simple Text Oriented Message protocol |
| TBD | To Be Defined |
| TMS | Traffic Management System |
| TSS | Traffic Scheduling System with TTM System and ATR functionality |
| TTM | Timetable Management System |
| UISR | EBI Screen User Interface Server and Message Router |
| URI | Uniform Resource Identification |
| XML | Extensible Markup Language |
| V&V | Verification and Validation |
| WildFly | An enterprise application server that implements Jakarta EE specification. |

## References

| **Reference** | **Document name** | **Document number** |
| --- | --- | --- |
|  |  |  |

# System igh Level High Level Configuration

The functionality of the EBI Screen 2000 ATS product is usually provided at two independent logical and functional levels:

* Central ATS (CATS) at the Operation Control Centre (OCC) and Backup Central ATS (BATS) at the Backup Control Centre (BCC) level
* Local ATS (LATS) at interlocking locations and depots

The CATS is responsible for online traffic management. Applications running in application servers connected to a trusted network are used to provide Centralized Traffic Control (CTC) and Traffic Management System (TMS) functions. Firewalls are used to protect the trusted network from potential threats from the non-trusted side of the network.

Some customers may require the provision of a spare CATS system in a Backup Control Centre (BCC) to be used in case of system failure of the main CATS in the OCC. Usually the functional and architectural requirements for CATS and BATS systems are the same.

LATS are typically used to provide supervision and maintenance related functions for one part of the railway, but they can serve as a fall-back control system in cases where CATS are not available, for example because of maintenance.

In some cases, customers may require an additional level of local control provided by the so-called Interlocking VDU Workstation for interlocking control and supervision in

case of unavailability of LATS or for maintenance purposes.



**Figure 1. High level functional nodes**

## Architecture Styles

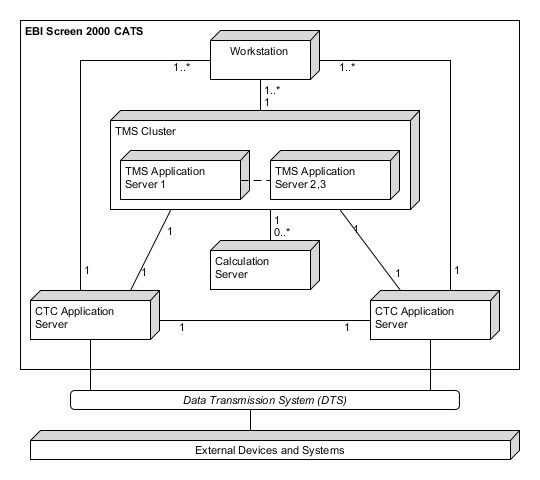
In the following chapters the abstract framework of the EBI Screen 2000 is described. These chapters contain a basic set of principles and high-level patterns that form the EBI Screen architecture.

### Architecture – Deployment

The typical architecture of CATS follows the client-server model where the functional decomposition of the system has been distributed into four logical nodes:

* Hot-standby CTC Servers
* Cluster of TMS Servers
* Pool of Calculation Servers
* Workstations.

The following figure shows the configuration of CATS with hot-standby CTC servers, clustered TMS servers, Traffic Calculation servers and client workstations. The ATS is connected to external devices and systems via Data Transmission System (DTS). Alternatively, the traffic Calculation servers’ applications can be hosted on TMS server computers.

­­­­

**Figure 2. CATS - Node configuration**

The typical architecture of LATS follows the client-server model where the functional decomposition of the system has been distributed into two logical nodes: CTC servers and Client Workstations. Usually the functional scope of LATS is a sub-set of CATS functionality. In small installations these two nodes can be physically hosted on the same computer.

CTC servers in LATS can be configured as a single server or a hot-standby server system. The following figure shows the typical configuration of LATS.



**Figure 3. LATS - Node configuration**

In some installations it is required to provide reduced TMS functionality at the local control level also. In this case the functional decomposition of the system is distributed into three logical nodes: CTC servers, a cluster of three TMS servers and Client Workstations. Virtual machines can be used to reduce the number of physical servers by hosting CTC server, TMS server and traffic calculation applications (DSolver) on the same physical computer.



**Figure 4. LATS – Node configuration with TMS functionality**

#### Application Servers

There are two types of application servers: CTC servers and TMS servers.

CTC application servers host a database and applications which are used to interface external devices and systems and to process and provide system runtime data to other subsystems. Usually CTC servers are arranged in a hot-standby configuration where the online server unit is responsible for controlling and monitoring external devices and systems while the other unit is a hot-standby ready to take over if the online server unit fails.

The TMS server hosts the Traffic Scheduling System which is responsible for providing traffic management functionality namely Timetable Management (TTM) and Automatic Traffic Regulation (ATR) functionality. Single point of failure is avoided by clustering three physical TMS servers. The Traffic Scheduling makes use of the calculation capacity of a number of solver applications (DSolver instances) which provide calculation services to the ATR to produce a minimal solution for a traffic regulation problem. Solver applications can be hosted on physical TMS servers or on separate Calculation servers.

TMS application servers also host data files, relational- and tabular SQL databases.

ATS applications can use message-based inter-process communication (IPC) mechanisms, databases or Web services to communicate one with another.

Virtual Machines can be used to reduce the number of physical servers.

#### Workstations

Workstations host the ATS User Interface (ATSUI) subsystem which constitutes the presentation layer (client operator workstations) of the ATS. Workstations are generic i.e. the functionality they provide depends on the profile of the current logged in ATS Operator. Redundancy is provided by the multiplicity of the generic workstations available in the control room.

#### Networking

All workstations and servers are connected to the same LAN. Dual LAN is used for higher availability.

Workstations can also be connected to a WAN, by means of routers or bridges connected to the EBI Screen LAN, the only requirement is that the IP protocol can be used over the WAN and LAN.

In installations where there is a OCC and a BCC control centre the corresponding ATS systems are connected to each other via fast Dual LAN connections. Physically 10G optical fibre is preferred because the amount of data transferred from the “Active” control centre to the “Inactive” one. Size, type and frequency of the data transmitted are project specific. The way in which OCC and BCC interact is project specific also (e.g. cross connection of workstations of the inactive control centre to the servers of the active control centre).

#### External Interfaces

Different types of communication lines and protocols are available for communication with external devices and third-party systems. The communication system can be connected to the application servers via serial lines or TCP/IP over Ethernet LAN/WAN.

A dual Local Area Network (LAN) connects ATS servers with the shared system-wide Data Transmission System (DTS) for conveying controls, indications and data to and from the remote sites (stations, traction Sub Stations and public information devices). DTS is used to connect with external systems e.g.: Interlocking and RATO (depending on specific project requirements)

OLE for Process Control (OPC) technology can be used to allow exchange of information between ATS and 3rd party external systems (e.g. Train Radio System, PIDS/PAS, SCADA, etc.) easily, securely and without costly, time-consuming software development.

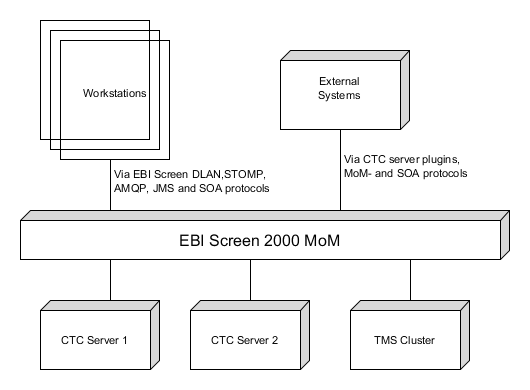
### Architecture – Inter-process Communication

ATS Inter-process Communication (IPC) is based on messages that are sent via Message Oriented Middleware (MoM), also known as Enterprise Service Bus (ESB).

In a logical level a common MoM is shared between CTC servers and TMS servers. It means that a single application can be located to any server and the messages can be sent transparently from one application to another. Implementation details and physical locations are hidden. MoM also integrates existing and new applications together in a loosely coupled way.

Applications in EBI Screen workstations can connect to EBI Screen servers via the EBI Screen specific communication protocols or via commonly used service-oriented protocols (SOA).

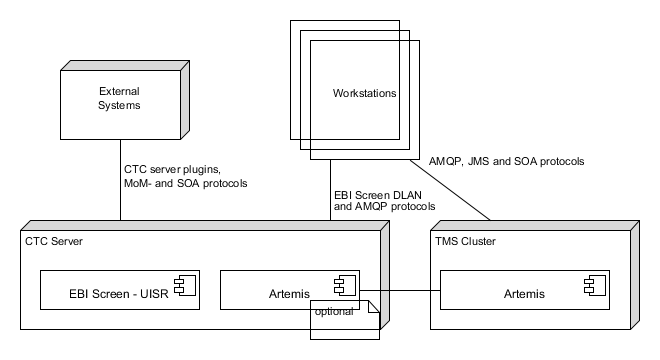
External systems can connect to the MoM via a plugin framework or SOA.



**Figure 5. EBI Screen 2000 - Logical MoM**

In more concrete level the logical MoM can be divided to EBI Screen proprietary- and third-party MoM. The main component of the EBI Screen proprietary MoM is a message router, also called User Interface Server Router (UISR). Apache ActiveMQ Artemis (Artemis), an example of Message Oriented Middleware, is third-party software which is integrated in the EBI Screen platform to provide connectivity between EBI Screen applications and towards external clients.

Both MoM types, depending on project requirements, can be used in CTC servers but TMS uses only Artemis.



**Figure 6. EBI Screen 2000 – MoM classification**

#### EBI Screen Proprietary MoM - UISR

The main component of the EBI Screen proprietary MoM is a message router, also called User Interface Server Router (UISR). It is a service installed in both CTC servers and in local CTC servers also. Its main task is to route:

* messages within CTC server applications
* messages between CTC server applications and client workstations
* messages between CTC servers in hot-standby configuration

Message routing can be enabled in two ways:

* An application can order the wanted messages from UISR (publish- subscribe pattern).
* Message routing can be done according to predefined routing tables.

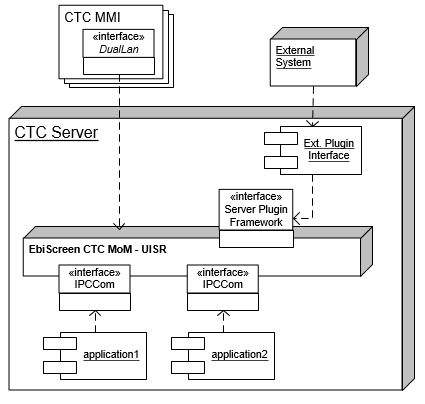
All messages can be secured with RIPEMD4-160 hash code in the application level.

Auxiliary libraries are used to give easy access to EBI Screen UISR MoM:

* IPCCom library: Used by server applications within a CTC server. Inter-process communication is based on named pipes. The message router creates these pipes, and applications connect to the pipes by using this library. The API provides support for the creation of a connection and the sending and receiving of messages.
* DualLan library: Inter-process communication library used by applications in a client workstation to communicate with applications in CTC servers. The communication protocol is UDP/IP and both LAN types (single or dual) are supported. An authentication mechanism is provided to validate client’s authority.
* Server Plugin Framework: provides connectivity and functional integration with external systems. It hides MoM implementation and provides also high-level platform services. It contains many auxiliary libraries that can be selected according to the needs of a specific external interface.
* A plugin concept that can be used to provide connectivity and functional integration with external systems. Specific DLLs e.g. TrainService.dll and the SubSysFrame -framework can be used for this purpose.

  
 **Figure 7. Plugin concept**

EBI Screen UISR MoM connectivity is illustrated in the following figure:



**Figure 8. Connectivity to UISR MoM**

#### Third-party MoM - Artemis

Apache ActiveMQ Artemis (Artemis), an example of Message Oriented Middleware, is third-party software which is integrated in the EBI Screen platform to provide connectivity between EBI Screen applications and towards external clients. A few of the Artemis properties are:

* asynchronous messaging system
  + configurable persistence and durability
  + point to point- and publish-subscribe patterns
  + transactions, message level load balancing
* very high performance
* multi-platform support
* multi-protocol support
* bridges and reliable routing also in wide area network

Artemis is used in CTC and TMS application servers. It supports many communication protocols that can be translated from one protocol to another via Artemis Core protocol.

Native Java applications can communicate with Artemis by JMS API. As in other protocols, it allows application to create, send and receive messages asynchronously. It defines a common set of interfaces and classes that allows applications to communicate with a message provider/broker, Artemis in this case. The broker routes a message to the destination where a consumer is waiting for the message to arrive. There are two destination types: queues for point-to-point communication and topics for publish-subscribe communication. These JMS specific destination types are mapped to Artemis addresses; the queue destination is implemented as ANYCAST routing type and the topic destination as MULTICAST routing type.

Java applications that follow the Jakarta EE specification, like WildFly and Payara, use typically Message-Driven Beans (MDB) for MoM communication. This type of bean normally acts as a JMS message listener, similarly to an event listener but listening to JMS messages instead of events. A client accesses a MDB through JMS by sending messages to the message destination for which the MDB class is the message listener. The MDB takes also care of several services like transactions, security and concurrency.

Service-Oriented Architecture (SOA) technologies can also be used because Java enterprise application server is an essential part of the TMS server. A Single Object Access Protocol (SOAP) web service specifies a self-explanatory interface to a consumer (client of a web service) by describing the used business methods and parameters. The consumer can use the service by any programming language because the consumer and a service are exchanging information using XML documents. Also, a service side can be implemented in any programming language.

Service oriented architecture can also be implemented using REST architecture style where every piece of information is resource and these resources are addresses using Uniform Resource Identifiers (URI).

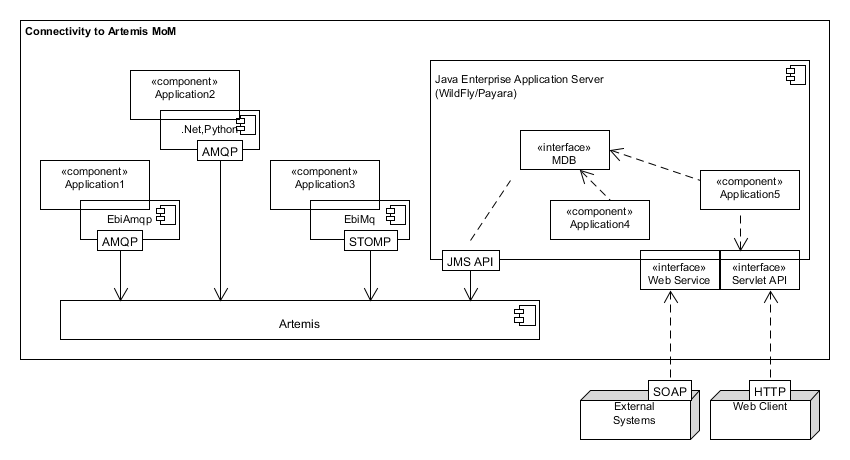
Both SOA techniques mentioned above are used with web clients and external (third-party) consumers to give exact interface to the services and hide the actual implementation of a service (loose coupling).

EBI Screen applications that are not written by Java can connect to Artemis by their own language specific libraries. In most cases the JMS style API is available and actual protocol implementation is based on AMQP or STOMP.

EBI Screen applications have been mainly implemented by C++ -programming language. Two separated libraries, EbiMq and EbiAmqp, have been created for easy access to Artemis. Those libraries hide an actual connectivity protocol implementation. Actually, the libraries hide the whole messaging broker. The main idea is that MoM can be replaced, if needed, without affecting the applications. Main features and properties of the libraries are:

* multi-threading, non-blocking connection management, message sending -and receiving
* support for message routing- and application specific properties
* message selection based on destination/addresses and properties
* automatic reconnection
* small footprint, easy to integrate and works in all modern operating systems

EbiMq library uses the STOMP 1.2 -protocol for Artemis connection and messaging. AMQP 1.0 -protocol is used in EbiAmqp library. One AMQP benefit is that Artemis does not perform any message conversion to any other protocols when sending AMQP and receiving AMQP. AMQP is also de facto standard in many other message brokers. EbiAmqp library supports TLS 1.2 also and it is possible to define more than one connection endpoints.

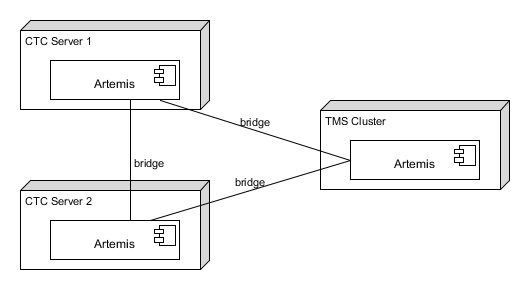


**Figure 9. Connectivity to Artemis MoM**

#### CTC – TMS IPC interoperability

Depending on project requirements the CTC and TMS can share a same Artemis. In that case the CTC applications connect to Artemis owned by TMS cluster.

More typical case is that there is also own Artemis standalone instance running in both CTC application servers. All applications connect to local Artemis that is controlled and supervised by CTC Watchdog as all other applications. All Artemis instances are bridged together but only relevant messages from one Artemis instance to another are routed through.



**Figure 10. Interoperability of Artemis instances**

### Architecture - Structure

EBI Screen 2000 software architecture focuses on the decomposition of the design into individual functional or logical components that expose well-defined communication interfaces containing methods, events, and properties. General architecture components hide the physical communication protocols and shared states of the used objects. This gives us many advantages like:

* reusable: same code can be used in many applications => reliability, verification
* replaceable: component can be easily replaced with another similar component
* extensible: current behaviour can be easily extended
* encapsulated: implementation details and internal processing is not shown outside
* environment free: components can operate different environment and context

Reusable components are physically located into libraries to ensure that a clear interface is given to applications. EBI Screen 2000 has many reusable components; some of them are listed on below:

* Communication related components: see chapter 2.1.2 Architecture – Inter-process Communication.
* ETrace.dll: Provides standard EBI Screen logging functionality by supporting application specific log files, windows event log and Linux system log. It also includes configuration for line pattern, automatic file rotation based on size and file purging.
* IndicationService.dll: Provides an interface for handling indications and commands between the EBI Screen platform and plugin applications.
* RailGraph.dll: Provides support for different railway geographies like track- and RSO networks:
  + network conversions (e.g. RSO position to track network)
  + navigation interfaces (e.g. find path to station/platform)
  + dynamic states of interlocking objects (e.g. signal states, point positions)
* Subsysframe.dll: Provides commonly used platform services like:
  + automatic supervision handling (watchdog interface)
  + interface to operation states (offline, standby and online) of an application server
  + message interface to MoM: message sending, receiving and ordering
  + timing services: absolute-, delay- and periodic timers
* TrainService.dll interface to common train services like
  + general train properties
  + train positions
  + train timetable
  + basic commands for describer association and modification
* DbCon.dll: Generic interface to EBI Screen Solid database; connection establishment, automatic refreshing

## Software Control Implementation

In EBI Screen 2000, control resides concurrently in several independent applications, each performing a specific task in an event-driven fashion. Events are implemented as one-way direct messages between applications.

## Application Control

The life cycles of the EBI Screen CTC applications are supervised and externally controlled by the EBI Screen watchdog (WD) application and with the aid of the operating systems. In TMS side, special application Pacemaker is used to make sure that all critical resources are available.

### CTC Application Server Control

CTC Application Servers can operate in one of the following states controlled by EBI Screen watchdog (WD) application:

* offline
* going standby
* standby
* going online
* online
* online – degraded

The state transition is shown in the following figure:



**Figure 11. State Diagram of CTC application servers**

In redundant configurations there can be only one CTC online server and one CTC standby server.

Both CTC application servers have a watchdog application, which supervises all EBI Screen applications running on this server and the state of the partner CTC application server. Also, the most important external third-party application frameworks like Artemis and WildFly are supervised.

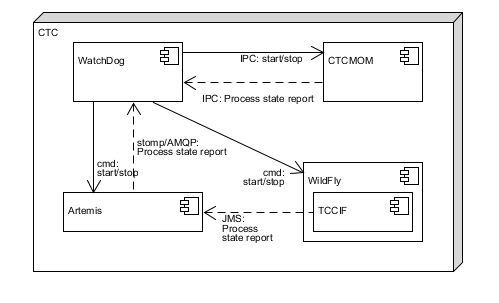
If the watchdog application of the online server finds that some critical application is not sending alive-messages or has exceeded its predefined memory limit, then the watchdog application sets the state of the application server to offline and informs the watchdog application of the standby server to change its state to online.

Also, if the communication link between the online and standby servers is broken down, the standby server changes its state to online.

If a single non-critical application fails only that application is restarted and there is no need for a server switch-over.

If an application that runs under Java enterprise application server, WildFly for example, does not fulfil the watchdog requirements the whole enterprise server must be restarted. The watchdog cannot know whether a problem is in enterprise server or application itself because they are sharing a same Java Virtual machine (JMV). Restarting can take time and for higher service availability the Java enterprise application server is typically defined as a critical resource. Two different applications that are not tightly coupled together, both are not needed to produce one service, must be run in separated Java enterprise application servers.

The following figure illustrates Watchdog functionality to control and supervise applications:



**Figure 12. An example about Watchdog supervising**

There is one relational database installed on each CTC application server. A database replication mechanism is used to update the database of the standby server from the database of the online server. Database role (master or slave) changes according to the role of the corresponding application server.

The state of a CTC application server can be manually changed by the System Engineer or automatically by the system.

Both CTC application servers have interfaces with external systems for different purposes. Commands to an external system are allowed only from the online server.

Each application shall be aware of single points of failure. A typical solution is to store stateful data of applications in the database. This is not allowed for the applications that are running in CTC server because the CTC database is used for generic application data that cannot change during a normal operation. Instead of that all stateful data is replicated from the online server to the standby server at the application level by using the EBI Screen MoM.

### TMS Application Server Control

There are no separate operational states in TMS servers. After starting up (which is activated from the watchdog of the CTC server) the TMS server is always ready for service. However, TMS servers are aware of the operational state of CTC servers and of their state changes. TMS servers are also aware of the state of all CTC applications. Each single application in a TMS server can decide whether this information is relevant or not.

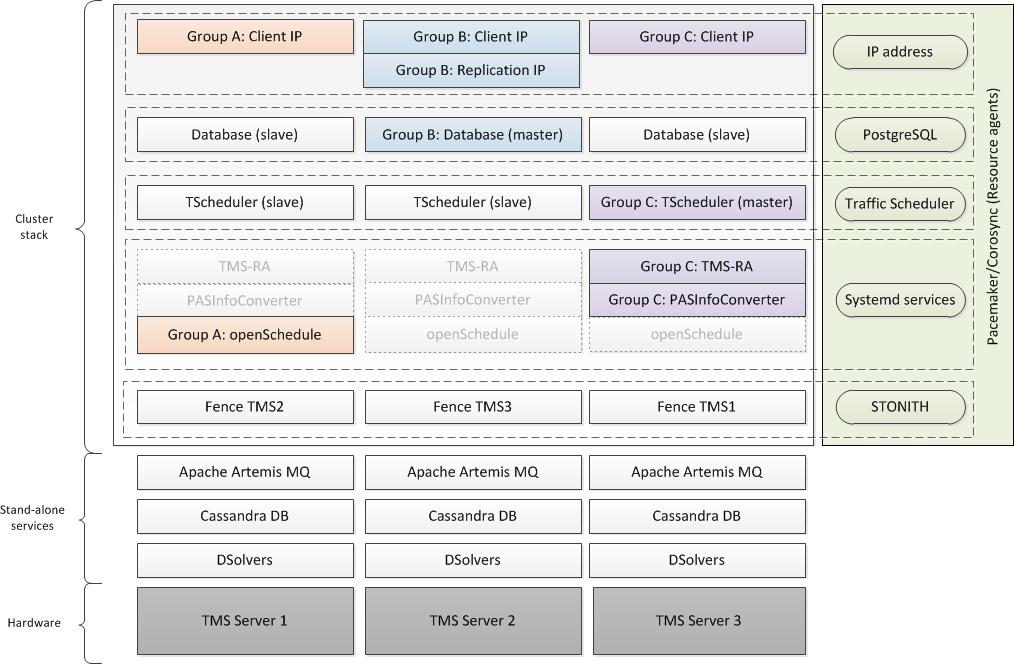
Single point of failure of TMS servers is avoided by clustering three physical computers (nodes) together. A special cluster resource manager (Pacemaker) is needed to make sure that all critical resources receive increased availability. If one resource fails in some physical computer, the cluster manager ensures that the service is available in other computer. Java enterprise environment and JMS message provider are also clustered. At the application level the Java bean instances are synchronized across all nodes, so that the client application reaches the same session state no matter which node serves the request. The TMS database is also clustered and running in all nodes.

#### TMS Cluster configuration

TMS servers can operate in a multi-node cluster setup to ensure service availability and redundancy. This functionality is provided by the following packages which must be installed in advance:

• pcs – is a Pacemaker configuration system

• fence-agents-all – fence agents



**Figure 13. High-level presentation of s-node cluster configuration**

Stand-alone services

Stand-alone services are running on each TMS server simultaneously. They are managed in a traditional way with systemctl command and are not part of the cluster setup.

Cluster service

Cluster stack contains TMS applications which are managed by Pacemaker/Corosync software suite.

In terms of cluster terminology, these applications are called cluster resources. Each type of cluster resource has its own resource agent, i.e. script or application which servers as API between the resource and Pacemaker service. A resource agent starts, stops, and monitors the resource state and reports it to the cluster SW. For Master/Slave types of resources, i.e. PostgreSQL database and TrafficScheduler, resource agent also has the operations of promote and demote.

TrafficScheduler resource agent is developed by BT and needs to be installed as a part of the TMS release. The rest of the agents are part of the RHEL High-Availability Add-on and are installed along with pcs and fence-agents-all packages.

Below is the list of services managed by the cluster software:

* Master/Slave resource set PostgreSQL database. There are three instances (one master and two slave) of the resource running simultaneously on all TMS servers. The replication IP address is used for synchronization between the instances. Client IP address is used for client connections.
* Master/Slave resource set TrafficScheduler. There are three instances (one master and two slaves) of the resource running simultaneously on all TMS servers. Client IP is used for client connections.
* Primitive resource TMS-RA. Pacemaker ensures that only one instance of the resource is active at any particular time.
* Primitive resource PASInfoConverter. Pacemaker ensures that only one instance of the resource is active at any particular time.
* Primitive resource OpenSchedule (as part of Payara application server). Pacemaker ensures that only one instance of the resource is active at any particular time. Client IP is used for client connections.

A group is a set of resources that must be running on the same TMS server. For example, it is not acceptable that TMS-RA from the group C is running on the TMS Server 2, while other resources from this group – on the TMS server 3. If one of the resources from the group fails on one server, the whole group is moved to the available online TMS server.

### Workstation Control

Normally workstations have links to both: to the online and to the standby CTC application servers but request data from and send commands to the online server only. Workstations are also linked to the TMS cluster.

The link to the CTC standby server is tested cyclically. If the CTC standby server becomes online server, automatically or manually, then the new online server informs all workstations, which are connected to it, about the switchover event. An event (the reason for a switchover) is registered into the event log and an acoustic alarm is given to the ATS Operator. The currently open displays (CTC views or log displays) are updated from the new online server.

The reconnection of a client workstation is managed by the client application itself. If the connection to the server is broken, the client polls both servers to find out which server is the online one.

### Backup Control Centre Control

In installations where there are OCC and BCC control centres, the CATS at the OCC and the BATS at the BCC are independent systems with their own set of ATS servers and workstations. During centralised mode of operation only one ATS system can be in active state at any moment,

Transfer of control from the active control centre to the inactive is always a manual operation supported by procedures established by the operation authority of the railway network.

Normally, data between the CATS and BATS is not synchronized online. Relevant data can be synchronized manually, e.g. before a pre-planned transfer, or automatically daily e.g. during non-commercial operation time at night.

Size, type and frequency of data transferred from the “Active” control system to the “Inactive” one is project specific. The way in which the CATS and BATS interact is project specific also (e.g. cross connection of workstations of the inactive control centre to the servers of the active control centre).



**Figure 14. CATS / BATS configuration**

# Architecture Design

## System Context

The following context diagram represents the more relevant external actors that interact with the ATS system:



**Figure 15. System context**

### ATS Operator

The Authority Management function of the ATS permits to define different profiles for the different users of the system. A user can have different roles if defined so. Following figure shows an example of the basic roles an ATS Operator could have.



**Figure 16. Operator role concept**

### Wayside Signalling

#### EBI Lock 950 R4

The CBI receives commands from the ATS and evaluates them with respect to traffic safety (interlocking) rules and the actual traffic situation. Only such commands that can be executed safely are executed.

The CBI monitors the railway infrastructure continuously and sends information regarding the state of traffic and the infrastructure back to the ATS.

The ATS interfaces with the EBI Lock computer-based Interlocking using UDP/IP protocol.

### Wayside ATO

This interface allows communication between ATS and Region ATO (RATO) to exchange information like train alarms, Skip/Hold commands, driving profiles to be transmitted to the trains, etc.

### Wayside TCC

This interface allows communication between ATS and INTERFLO 150 TCC to exchange information like train information wayside object statuses (e.g. points, detailers) alarms, TSR, possessions.

### Video Walls

This interface allows transfer of data from ATS system to an external projection system on a video wall or similar system (if existing). By means of this interface, a live representation of the signalling system and the running trains can be presented on a large screen.

ATS can interface with video wall systems either via the Large Screen Client (LSC) component or via OPC-based interface.

### Master Clock

The primary source of time reference of the ATC system usually is an external master clock acting as NTP server. The master clock is connected to the ATS system via network.

### Project-specific Internal Systems

#### CITYFLO 150 Onboard Subsystem

Through train to wayside bi-directional radio communication system, ATS sends to the onboard system punctuality information, timetable ID, front obstacle type and distance, front road junction priority status, next/terminal station, etc. The onboard system sends to ATS the train ID, train operation status, onboard equipment health status and alarms, road junction priority control command, GPS/BD messages etc.

#### EBI Lock 400

EBILock400 constitutes the mainline points control subsystem and the road junction signalling control subsystem in some CITYFLO 150 solutions.

### Project-specific External Systems

These can be interfaces with 3rd party systems like passenger information, train radio, CCTV, SCADA, etc. These types of interfaces are usually implemented either as OPC-based or message-based interfaces.

Interface with external systems may be performed by means of a gateway PLC installed in the control centre or in local stations in the case of local ATS systems. The gateway acts as a firewall (to protect the signalling DTS) as well as a data format/protocol converter.

## System Decomposition

The high level subsystem decomposition of EBI Screen 2000 is shown in the following figure:



**Figure 17. High level subsystem decomposition**

Subsystem responsibilities are shown in the following table:

|  |  |
| --- | --- |
| Ebibase | Provides system management, basic supervision and control functionality, and services to other Subsystems. Part of EBI Screen MoM. |
| Ebicore | EBI Screen basic CTC functionality. |
| Wayside Interfaces | Provides interfaces to wayside signalling related equipment and systems. |
| Onboard Interfaces  (via wayside ATO) | Provides interfaces to wayside ATO related equipment onboard of vehicles. |
| External Interfaces | Interface to existing external systems e.g.:   * Fleet- and Crew scheduling and dispatching functions. * Passenger Information Systems * SCADA systems * Radio systems |
| Traffic Scheduling System | Provides timetable management and optimization functionality. |
| Training and Simulation | Simulates and analyses system performance and operation. |
| Tools and Simulators | Includes all the tools necessary for the engineering and testing of the system. |

## Subsystem Division and Referral

In this chapter are described EBI Screen 2000 subsystems. More detailed descriptions are in relevant subsystem requirement and architecture documents.

### Ebibase

The Ebibase subsystem includes following subsystems:



**Figure 18. Ebibase subsystems**

* Supervision and changeover
  + Watchdog functionality; see chapter 2.3 Application Control.
* System Management
  + Includes tools for administrators to perform monitoring, control and maintenance actions, like
    - monitoring and changing operation mode of servers
    - monitoring and changing switchover mode
    - monitoring resource usage of processes
    - stopping and starting processes
    - generating reports
    - managing normal/daylight saving time changes
* Inter-process communication
  + Includes User Interface Message Router (UISR) and auxiliary communications libraries, see chapter 2.1.2 Architecture – Inter-process Communication.
* Licensing
  + Protects the system software from unauthorized usage.

### Ebicore

The Ebicore subsystem includes the following CTC-related subsystems:



**Figure 19. Ebicore subsystems**

* Authority Management
  + Manages authority areas and operator privileges
* Interlocking Command and Data Handler
  + Processes the data from process interfaces (interlocking, RTU’s) and delivers Event and Alarm Logging system
  + Controls the command processing from User Interfaces to process interfaces.
* Event and Alarm management
  + alarm supervision, acknowledgment handling, cancelling
  + alarm filtering
  + alarms by authority area and operator role
  + audible alarms by authority area and operator role
  + storing of events for long time history
  + display of events to operator
  + filtering of events
  + export of events to csv-files
* User Interface
  + Interface between the operator and the system.
  + Consists of main user-interface application and multiple AddOn applications
* Automatons
  + Runs automaton scripts by using Python interpreter and user interface addon which provides command and monitoring interface to the automaton functionality.
  + Automatons are used to program and execute frequently used command sequences. These can be supplemented with various conditions; to start a particular sequence, to wait within a sequence, etc. the condition specified may be the occupancy of a track circuit, the state of a signal and so on.
* Persistent Data provides the generic application data required by the different applications:
  + interlocking adaptation
  + ATO adaptation
  + track- and segment geographies
  + application configuration data
* Playback system logs all the system events and operator commands for later viewing:
  + replays the pictures, events and alarms in separate offline system
  + replay speed control; faster, slower and pause
* Route Server
  + Creates routes and train driving information automatically according to instructions from TMS and train movement. Also, manual commands are supported.
* Train Tracking System
  + Identifies all trains within a system area and maintains train position information of these trains according to train detection. It gives also train related services to other subsystems.
  + Communicates with interface applications / external system by using EbiMq; transferring train related information (e.g. train positions) and commands
* Mail, Note and Reminder
  + Operator notes
  + Mails between operators
  + Reminders
* Restriction Server
  + Manages temporary speed restrictions and possession
  + Processes indications from the interface applications
  + Gives indications to the user interfaces
  + Receives commands from the user interface
  + Gives commands via interface applications to the external systems (CITYFLO 650 RATO, INTERFLO 150 TCC)
* CTC MoM
  + Provides MoM-related functionality for the CTC subsystem
  + Interface to CMM messaging for wayside objects.

### Interfaces

#### Wayside Interfaces

EBI Screen 2000 system communicates with wayside objects via electronic interlocking. A communication with electronic interlocking EBI Lock 950 happens via EBI Lock 950 R4 Plugin process. These wayside specific interfaces communicate with a generic Command and Data Handler that is standard interface to other EBI Screen applications.

The project specific interfaces can be handled via OPC (open connectivity via open standards) interface implemented in ATS PIEDI or IEC60870 transmission protocol implemented in a process interface PI104.

PiEdi (EBi Screen Plug-in) is used for the message protocol which communicates between EBi Screen and with all of the configured external data sources (EDI). The relationship between the PiEdi and EDI is a Client / Server relationship, and the current implementation is an Ethernet TCP/IP based. PiEdi is the Server and EDI is the Client.

The Wayside Interfaces subsystem includes the following subsystems:



**Figure 20. Wayside Interface subsystems**

#### Onboard Interfaces

The Onboard Interfaces subsystem includes following subsystems:



**Figure 21. Onboard subsystems**

* CITYFLO 650 ATO Plugin (aka RATO Server)
* Implements the interface to CITYFLO 650 Wayside ATO. This is not a direct interface, the actual interface to onboard takes place between WATO and VATO.
* Communicates with the WATO, supervises the communication
* Handles the conversions between different geographical models
* Receives train, station, equipment status from the WATO and sends corresponding requests to WATO according to request from user interface and other subsystems
* Sends departure instructions to WATO according to train position and definitions in the timetable
* Provides CBTC train information to the Train Server

#### Combined Onboard / wayside Interfaces



* CITYFLO 650 ATO Interface
* Implements the interface to CITYFLO 650 Wayside ATO for Generic Product
* Communicates with the WATO, supervises the communication
* Handles the conversions between different geographical models
* Receives train, station, equipment status from the WATO and sends corresponding requests to WATO according to request from user interface and other subsystems
* Sends departure instructions to WATO according to train position and definitions in the timetable
* Exchanges CBTC train information (indications and commands) with the Train Server via Artemis MoM.
* Exchanges wayside indications and commands with the Control and Data Handler via Artemis MoM.
* Exchanges Temporary speed restrictions with Restriction server via Artemis MoM.
* Message structures delivered via Artemis MoM are according to CMM-model.
* INTERFLO 150 TCC interface
* Implements the interface to INTERFLO 150 TCC.
* Communicates with the TCC, supervises the communication
* Receives train, station, equipment status from the TCC and sends corresponding requests to TCC according to request from user interface and other subsystems
* Exchanges train information (indications and commands) with the Train Server via Artemis MoM
* Exchanges wayside indications and commands with the Control and Data Handler via Artemis MoM
* Exchanges Temporary speed restrictions with Restriction server via Artemis MoM
* Message structures delivered via Artemis MoM are according to CMM-model

### Tools and Simulators

The Tools and Simulators subsystem includes the following subsystems:



**Figure 22. Tools Subsystems**

**Engineering tools:**

* Symbol Type Designer
  + Used to define states for user-interface objects
* Symbol Builder
  + Used to define how states defined with Symbol Type Designer are visualized
* View Builder
  + Used to define how symbols created with Symbol Builder are connected to each other to form a view.
* Symbol Manager
  + Used to generate a document of symbols, and to migrate symbols from different sources
* Site Data Management Tool
  + Form-based tool used to maintain site data.
* pgAdmin
  + graphical user interface administration tool for PostgreSQL
* PostGis
  + TMS form-based database maintenance tool
* Python IDE
  + Used to edit automaton scripts.
* External XML Editor
  + Used to edit files in XML format
* Data Import Tools
  + Map Import tool for Regional data: e.g. geographical RSO (Region, Segment, Offset) model with connection to Interlocking data, vehicle related data
  + Import tool for EBI Tool data: e.g. Interlocking commands and route definitions

**Installation and Configuration tools:**

* Setup Tool
  + Setup program used for installing
* Configuration Check Tool
  + Lists version numbers and calculates MD5 checksum for an EBI Screen 2000 installation
* License Manager
  + Used to manage license information

**Testing tools:**

* Train simulator (RATOSimulator)
  + Runs trains according to the routes generated by the infrastructure simulator.
* Object State Viewer
  + User-interface addon allowing user to view and modify internal states of system objects
* PI Message Player
  + Reads message log files recorded by different Process Interface applications and feeds the messages to the Command and Data Handler.

### Traffic Scheduling System

#### Subsystems in TMS Server

The Traffic Scheduling System in the TMS server includes the following components and subsystems:



**Figure 23. Traffic Scheduling System components in TMS server**

* Traffic regulation module is responsible for optimizing timetables online to minimize predefined cost function.
* Conflict detection module is responsible for notifying operator for future conflicts which have specified not to be resolved automatically.
* Conflict resolution module solves the conflicts for regulation module.
* Service and crew data importer are responsible for importing static crew and service timetables.
* Traffic statistics module collects driving times, dwell times etc.
* Fleet and stabling manager keeps track of train positions and alerts if resources do not match the timetables.
* Schedule manager is responsible for making changes to the timetables with the help of other modules.
* Cluster resource manager is used to manage resources to provide high availability of the services.
* Crew manager is responsible for crew information and is used for example by schedule manager when trains are reformed etc.
* Web server is a HTTP-server for providing services to user interfaces applications.
* Traffic events and alarms is monitoring and predicting future traffic and providing traffic notifications.
* EBI Screen messaging is the so called Ebibase-subsystem.
* Messaging middleware provides communication between components implemented in different programming environments.
* persistent storage for the scheduling services:
  + planned-, estimated- and actual timetables
  + traffic statistics
  + traffic notifications
  + train and vehicle properties
  + train and vehicle positions
* persistent storage for the Geographic Information Systems (GIS)-, crew- and fleet data

#### Offline Simulation

An ATR simulator is used to validate the configuration of the ATR for the different traffic situations. These situations can be simulated by running predefined traffic scenarios. A simulation output can be verified for example by using a train graph.

The purpose of simulator is:

* simulating system performance and system operation
* supporting different simulation scenarios
* analysing system performance, the impact of the track configuration, effects of different signalling systems, headway and line capacities, the impact of system delays and failures, and the operational safety and robustness of the system

### ATS User Interface Subsystem (ATSUI)

#### Context

The following context diagram represents the relevant external actors that interact with the ATSUI sub-system:



**Figure 24. ATSUI context**

#### Decomposition

The ATS User Interface consists of the following components:

* CTC User Interface (CTCUI).
* Traffic Scheduler User Interface (TSUI).
* Alarm Management User Interface (AMUI).
* Auxiliary Project-specific User Interfaces.

The components listed above are implemented as separate Windows applications.

**Figure 25. ATSUI components**

#### Configuration

The following figure shows the configuration of the ATSUI in an ATS Client Workstation with connections to CTC server and TMS Cluster. A dual LAN is used for connections between nodes.



**Figure 26 ATSUI Configuration**

An Artemis server is used as a MOM for inter-process communications between the core applications (CTCUI, TSUI, AMUI and ClientMan) of the ATSUI.

AMQP -protocol and JMS API are used by client applications to interact with Artemis MoM.

CTCUI uses the EBI Screen proprietary dual LAN protocol (DLAN.dll) for communication with the UISR message routers in CTC servers.

TSUI is a client of the TMS applications in the TMS cluster.

The Stomp- and AMQP -protocols are used for messaging interoperability between specific CTC server applications and the Artemis MoM in the TMS cluster.

The communication between browser-based user interface web applications with the Web Server components in CTC or TMS servers takes place using HTTP.

#### Interfaces

Following interfaces have been identified:

|  |  |
| --- | --- |
| Actor | Interface |
| ATS Operator | Keyboard, mouse, monitor, speakers, video wall |
| TMS Server | Artemis MoM |
| CTC Server processes (CTCUI Server, EH, AUTS, WD, TS, AUTM, RATOS, AUTS) | DLAN |

##### CTC User Interface (CTCUI)

CTCUI is the main user interface for CTC functionality. They provide information and support user interaction through:

* CTC views
* CTC Track Layout Views, where each CTC View is a track layout picture with zooming and panning functionality. At application data design phase each CTC View can be designed to contain either a detailed layout of an area or an overview of a large area with limited amount of data on it.
* System Status View, showing the status of the main software and hardware components of the system.
* Maintenance Views. These are maintenance-oriented views showing the status of specific monitored objects.
* Playback User Interface for selection and playing of playback files.
* Train Describer functions: Set, Delete, Move, Exchange.
* Automaton User Interface

Figure below depicts the main functional components of the CTCUI:



**Figure 27. CTCUI components**

##### CTCUI Architecture

CTCUI consists of following modules:

|  |  |
| --- | --- |
| Module | Description |
| UIC.exe | The main assembly of the CTC User Interface. |
| CommandParser.dll | Command Parser |
| ebilkib8u.dll | In the first stage this can be a C# “encapsulation” of the legacy C++ module shared by legacy UIC, VB, SB, Sysman, WebUI (currently not part of EBI Screen 3.7). The module needs new IF to be accessible from C# applications. Might be used also by ClientMan for visualization of object statuses. The C# encapsulation shall have interfaces compatible with the C# version of Ebilib being developed for the Italian user interface. |
| dlanlib8.dll | Interface to CTC server message router (UisrService) |

#### Event/Alarm Management User Interface (AMUI) Applications

AMUI constitutes the main user interface for event and alarm management functionality. In a standard configuration, the Alarm Management User Interface consists of summary alarm indicators and the user interaction function provided by the generic Alarm and Event Handling applications of EBI Screen. In the future and if required the generic Alarm and Event Handling of EBI Screen could be substituted by another non-EBI Screen component like MDC.



**Figure 28. AMUI components**

##### AMUI Architecture

AMUI consists of following modules:

|  |  |
| --- | --- |
| Module | Description |
| event.dll | Event and alarm lists |

#### Traffic Scheduler User Interface

The Traffic Scheduling System in TMS workstation includes following subsystems:



**Figure 29. TMS Client components**

Figure above depicts the modules running in the workstation.

* Timetable WebUI is browser-based access to timetables etc.
* Timetable UI is non browser-based view to timetable data.
* Train graph shows the predicted traffic situation in graph format.
* Traffic dashboard provides quick overview of traffic situation
* Traffic view is used to graphically change timetables and monitor traffic in the track network.
* Notifications UI show traffic related events and alarms

## Implementation Rules

### Operating System

There are used four different operating systems in EBI Screen 2000.

* Windows Server 2012 (CTC Application Server)
* Windows 7 Professional, 64 bits (Client Workstation, LCS)
* Linux Centos (Traffic Calculation Server)
* Red Hat EL7 (TMS Application Server)

### Programming Languages

The following languages and IDEs are used:

* C++ by using Microsoft Visual Studio- and GCC compilers
* C# by using Microsoft Visual Studio
* Java via NetBeans

### Internationalization

Following rules apply:

* English is used for internal messages and application logs
* CTC user interfaces use EBI Screen Multilanguage Support Package in C++ applications
* UTF16 or UTF8 is used. UTF16 is a default implementation if decoding has not explicitly specified.

### Association Implementation

Not Applicable.

# Architecture Components

The subsystem components are allocated to the following nodes and the configuration items are identified (an asterisk (\*) in the configuration item name means that there are multiple variations of the same item):

| **SUBSYSTEM** | **RELATED SUBSYSTEM** | **CONFIGURATION ITEM** | **NODE** | **NOTES** |
| --- | --- | --- | --- | --- |
| Ebibase | Inter-process Communication | dlanlib8.dll | -Client Workstation | DLAN protocol library |
| Ebibase | Inter-process Communication | mdlanlib.dll | -Client Workstation | Managed DLAN protocol library |
| Ebibase | Inter-process Communication | EbiDNS.exe | CTC Application Server  -Application Servers | EBI Screen Domain Name Server |
| Ebibase | Inter-process Communication | ipccom\*.dll | -Client workstation -Application Servers | Inter-process Communication Library |
| Ebibase | Various | simplelib.dll | Client workstation -Application Servers | Thread library |
| Ebibase | Inter-process Communication | uisrservice64.exe | Application servers | Message Router |
| Ebibase | Supervision and Changeover | ClientWD.exe | Client workstation | User Interface Client Watchdog |
| Ebibase | Supervision and Changeover | WDS\_DE.exe | CTC Application server | Watchdog (Service version) |
| Ebibase | System Management | sysman.exe | -Client workstation  -Application servers | System Manager |
| Ebicore | Various | DbCon.dll | CTC Application Server | Common library for Solid database connection |
| Ebicore | CTC MoM | ctcmom.exe | CTC Application Server | CTC MOM interface server |
| Ebicore | CTC MoM | EbiMq.dll | CTC Application Server | EBI Screen message queue library, base on STOMP -protocol |
| Ebicore | CTC MoM | EbiAmqp.dll | CTC Application Server, Client Workstation | EBI Screen message queue library, based on AMQ -protocol |
| Ebicore | CTC MoM | ctcmomengine.dll | CTC Application Server | CTC MOM engine library |
| Ebicore | CTC MoM | cmm-command-indication.dll | CTC Application Server | CMM command indication |
| Ebicore | CTC MoM | tms-e2k-converter.dll | CTC Application Server | TMS interface specific converter |
| Ebicore | CTC MoM | valladodid-leon-burgos.dll | CTC Application Server | Spain HSL project specific converter |
| Ebicore | Various | IpcForServer\*.dll | CTC Application Server | Common library to give access to IPC for SubsysFrame based applications |
| Ebicore | Various | IndicationService.dll | CTC Application Server | interface for handling indications and commands between the EBI Screen platform and plugin applications |
| Ebicore | Authority  Management | Autm.exe | CTC Application Server | Authority Handler |
| Ebicore | Automatons | Autc.dll | Client Workstation | Automaton Client |
| Ebicore | Automatons | AUTS.exe | CTC Application Server | Automaton Server |
| Ebicore | Event Logging and Alarm Handling | EH.exe | CTC Application Server | Event and alarm management |
| Ebicore | User Interface | event.dll | Client workstation | CTC Event & Alarm User Interface |
| Ebicore | Event Logging and Alarm Handling | EventExport.exe | CTC Application Server | Exports events to csv.files |
| Ebicore | Interlocking Command and Indication Handling | cdh.exe | CTC Application Server | Control and Data Handler |
| Ebicore | MDC Interface | PISdi.exe | CTC Application Server | SDI (MDC) Interface |
| Ebicore | Mail and note | MailServer.exe | CTC Application Server | Mail, note and reminder server |
| Ebicore | Mail and note | MailAddOn.exe | Client workstation | Mail, note and reminder user interface |
| Ebicore | Various | RailGraphX.dll | CTC Application server | Rail Graph Library |
| Ebicore | Various | SubSysFrameX.dll | CTC Application server | Subsystem Frame Library |
| Ebicore | Various | ETrace.dll | CTC Application server | Tracing Library |
| Ebicore | Various | TraceService.exe | CTC Application server | Tracing service |
| Ebicore | Route Setting | Ros.exe | CTC Application server | Routing Automation |
| Ebicore | Playback | VSPlayer.exe | CTC Application server | Player for playback data |
| Ebicore | Playback | VSLog2.exe | CTC Application server | Logger for playback data |
| Ebicore | Playback | UIC.exe | Client workstation | Integrated viewer for playback data |
| Ebicore | Persistent Data | Solid database | CTC Application server | database for CTC data |
| Ebicore | Train Tracking System | Tds.exe | CTC Application server | Train Describer System |
| Ebicore | Train Tracking System | TrainAddon.dll | Client workstation | Train Tracking System user interface |
| Ebicore | Train Tracking System | TS.exe | CTC Application server | Train data handler |
| Ebicore | Train Tracking System | TrainService.dll | CTC Application server | Library for dynamic train data. |
| Ebicore | Train Tracking System | TrainAddon.dll | Client Workstation | AddOn for train commands and data |
| Ebicore | Inter-process Communication | MsgRepServer.exe | CTC Application server | Replicates IPC messages between two EBI Screen systems, server part |
| Ebicore | Inter-process Communication | MsgRepClient.exe | CTC Application server | Replicates IPC messages between two EBI Screen systems, client part |
| Ebicore | User Interface | CommandParser.dll | Client workstation | Command Parser |
| Ebicore | User Interface | UIC.exe | Client workstation | User Interface Client |
| Ebicore | User Interface | ebilib8u.dll | Client workstation (Engineer) | EBI Screen Graphic Library |
| Onboard Interfaces | CITYFLO 650 ATO Plugins | RatoServer.exe | CTC Application Server | CITYFLO 650 Full RATO server |
| Ebicore | CITYFLO 650 ATO Plugins | WaysideSup.exe | CTC Application Server | Supervises wayside and controls platform holds. |
| Onboard Interfaces | TCC Interface | TCCInterface | CTC Application Server | INTERFLO 150 TCC Interface |
| Onboard Interfaces | CITYFLO 650 ATO Plugins | RATOInterface | CTC Application Server | CITYFLO 650 Generic Product RATO Interface |
| Tools | Engineering Tools | SM.exe | Client workstation (Engineer | Symbol Manager |
| Tools | Engineering Tools | SMT.exe | -Client workstation (Engineer)  -CTC Application Server | Site Management Tool |
| Tools | Engineering Tools | TLV.jar | Client workstation (Engineer) | Site data verifier |
| Tools | Engineering Tools | STD.exe | Client workstation (Engineer) | Symbol Type Designer |
| Tools | Engineering Tools | VB.exe | Engineering workstation | View Builder |
| Tools | Installation Tools | lmanager.exe | CTC Application Server | License Manager |
| Tools | Testing Tools | IPCMonitor.exe | CTC Application Server | Monitors IPC |
| Tools | Testing Tools | TrainingSim.exe | CTC Application server | Infrastructure Simulator |
| Tools | Testing Tools | RatoSimulator.exe | CTC Application server | 650 Rato and train simulator |
| Tools | Testing Tools | PiMsgPlayer.exe | Application server | Process Interface, Message Player |
| Tools | Testing Tools | RtDbVw.exe | Application server | Real Time Database Viewer. |
| Tools | Testing Tools | ObjStatAddOn.dll | Client workstation (Engineer) | Object State Viewer |
| Traffic Scheduling System | Train List Client | TrainListClient | Client workstation | Traffic Scheduler user interface |
| Traffic Scheduling System | Train List Client | OpenSchedule | Client workstation | Web Application for timetable management |
| Traffic Scheduling System | -Traffic Regulation  - conflict detection and resolution  -traffic statistic  -traffic events and alarms  -crew manager  -fleet and stabling  -service and crew | TrafficScheduler | TMS Application Server | Traffic Scheduling Server application |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | libtmsodbc++ | TMS Application Server | Traffic Scheduler libraries |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | tschedragent | TMS Application Server | Traffic Scheduler resource agent for the cluster setup |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | tms\_ra | TMS Application Server | Automatic route setting. |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | tmsstatusinformer | TMS Application Server | Supervisor of TMSapplications |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | pasinfoconverter | TMS Application Server | TMS Passenger information system interface |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | DSolver | Traffic Calculation Server/TMS Application Server | Distributed Optimization Problem Solver |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | HornetQLogger | TMS Application Server | HornetQ (Artemis) Message Logger |
| Traffic Scheduling System | Persistent Data | postgresql | TMS Application Server | Open source database management system |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | Apache ActiveMQ Artemis | TMS Application Server | Open source asynchronous messaging system |
| Traffic Scheduling System | Persistent Data | cassandra | TMS Application Server | Open source distributed database management system |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | Payara | TMS Application Server | Open source application server for the Jakarta EE platform |
| Traffic Scheduling System | Optimized Automatic Traffic Regulation | Pacemaker | TMS Application Server | Component to configure cluster infrastructure |
| Tools | Optimized Automatic Traffic Regulation | pgAdmin | TMS Application Server | Open source graphical user interface administration tool for postgreSQL |
| Tools | Optimized Automatic Traffic Regulation | PostGIS | TMS workstation | Open source software program that adds support for geographic objects to the PostgreSQL database. |
| Wayside Interfaces | Electronic Interlocking Plugins | PiR4.exe | Application server | Process Interface, Ebilock 950, R4 |
| Wayside Interfaces | Project Specific Interfaces | PIEdi.exe  Edi.exe | CTC Application server | Process Interface to OPC |
| Wayside Interfaces | Project Specific Interfaces | PICmms.exe | CTC Application server | Sends alarms to external system (PLP specific) |

# Interfaces

There are two types of interfaces: external and internal.

## External Interfaces

External systems can be physically connected via the serial lines or Ethernet interfaces, see more about connectivity of the external actors from chapter 3.1 System Context.

## Internal Interfaces

The major internal interfaces and connectivity have been described in chapter 3.1 System Context.

# ARCHITECTURE MANAGEMENT

## Memory Management

The following memory management issues are recognized:

* object related attributes and parameters are stored into the relational database
* the relational database is accessed through ODBC- or JDBC interface
* Persistent data is always handled by a server-side application, a client application does not have a direct link to the persistent data (three –tier architecture)
* common libraries for database usage

## Handling of Global Resources

The following shared resources and common access mechanisms to them are identified:

* MoM access via common libraries
* distributed processing for traffic calculation
* common locking mechanism for accessing shared memory
* access to EBI Screen Registry handled via SubSysFrame services
* common timing services via SubSysFrame
* usage of common libraries for database connection, e.g. DbCon.dll

## Identification of Concurrence

The main concurrent categories are:

* concurrent co-operating applications are running in application servers
* distributed processing for traffic calculation
* persistent data management by database transactions with replication

## Selection of Architectural Framework

The EBI Screen 2000 is based on the following common subsystem forms and architectures:

* a real time system with continuous data transformation as inputs change
* interactive user interface usually runs in dedicated computers, one per user

## Handling of Boundary Conditions

### Initialisation

In the initialisation phase, the following is done:

* most of the processes use the SubSysFrame component for initialisation of message/IPC communication, database connection and watchdog activation/reporting
* the SubSysFrame style initialisation mechanism is used to initialise all server specific subsystems

### Termination

All allocated resources are released in termination phase:

* all dynamically created objects and treads are destroyed and deleted
* dynamically allocated memory is released
* all reserved external resources are released
* process termination is reported to Watchdog (SubSysFrame)

# Architecture Safety

## Safety Strategy

EBI Screen 2000 generic product software is assured using SIL2 methods, however this does not make it a SIL2 system and it should not be described as such as no safety functions have been identified and consequently there are no defined safety requirements.

In most applications none of EBI Screen’s functions require to have a SIL above 0.

Where a few particular functions do have a safety integrity requirement these can be delivered using such techniques as ‘click and confirm’ which use feedback from the SIL 4 interlocking to confirm correct operation.

## Safety Techniques

Hardware – Only COTS hardware is used in the EBI Screen. Safety aspects are mainly taken care of by the application environment and the application SW. The use of ECC memory used in the COTS CPU platforms is taken into account as mitigation against random memory faults. Redundant hot-standby hardware is used to ensure availability.

Operating system – Standard COTS operating systems are used. Safety aspects are taken care of by the application environment and the application SW.

Application environment – The application environment is re-used.

Application SW – Developed in conformance to EN 50128.

Engineering Process – The engineering process for EBI Screen is based on the general engineering process. It includes several steps of verification at both engineering and customer site. The basic process has been applied for several years on many applications of the EBI Screen product thus giving assurance of correctness and adequacy.

Installation process ensures correct installation of the EBI Screen.

Maintenance procedures which ensures correct maintenance and configuration of the EBI Screen.

## Failure

The general handling mechanism is as follows:

In the Application Server side:

* Exception mechanisms are used as much as possible.
* Internal logs used to inform about problems
* Information about problems is written to application log and the process is restarted by the Watchdog or cluster resource manager.
* Applications are monitored by the Watchdog process and restarted if terminated abnormally.

In the Client side:

* Confirmation Message Box for application exit situations
* Automatic reconnection to the actual online server in case of server change-over.
* Operational redundancy of client workstation (auto configuration according to user privileges)

# Architecture Performance

## Scheduling Capacity

Common scheduling tasks:

* Backup tasks, carried out by the Scheduled Task Manager of the operating system
* Additional scheduling provided by the database engine e.g. for daily database backup
* Summer/wintertime change

## Estimating Hardware Resource Requirements

Not applicable.

## Hardware-Software Trade-offs

Not applicable.

## Throughput

Not applicable.

## Storage

Dynamic test and bound checking

## Trade-Off Priorities

Not applicable.

# Architecture Reliability

High availability is achieved by using:

* redundant hot-standby configuration for CTC application servers
* clustering for TMS servers
* distributed processing for traffic scheduling calculations
* database replication
* operational redundancy based on multiplicity of universal operator workstations
* dual LAN with network teaming

In case of failure in TCP/IP communication, current connection is closed and a new connection is started.

# Architecture Maintainability

## Test Features

Testing tools:

* Infrastructure Simulator (SIM).
* RATO/Train simulator (RATOSIM)
* Interlocking Simulator
* Object State Viewer (ObjState addon)
* IPC Message Player
* View Builder’s *Event Codes* function
* SMT’s *Data Analyser* function

## Diagnostics and Fault Reporting

Diagnostics and fault reporting tools:

* Internal application logs
* Message logs
* Playback
* Windows Event Viewer
* Linux syslog
* ObjState addon to observe and simulate the internal state of objects
* application specific log display or log file
* diagnostic counters are sent to EBI Screen core
* Watchdog-based supervision
* EBI Screen Reporter tool and Error Reporting Form (error reporting template)

End of document.