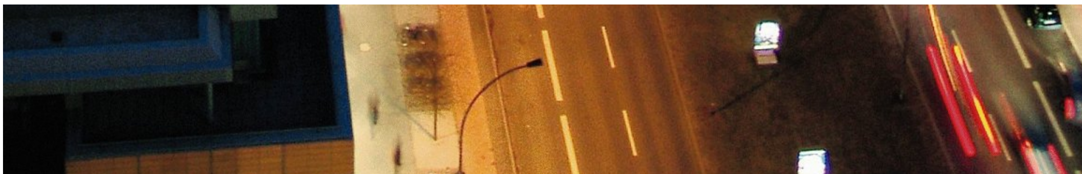
Sitraffic sX DWS

Sitraffic sX Control Model V1.0
A001

Intelligent Traffic Systems

SIEMENS



Contents

List of Figures

References

No.	Document	Content	Where to find?
[1]	Sitraffic sX DWS Installation Guide		Sitraffic_sX_DWS_Installation_Guide_en.pdf
[2]	Controller GUI User Guide		<installdir>\dws\doc\Sitraffic_sX_WebGui_Manual_en.pdf
[3]	Sitraffic smartCore User Guide		<installdir>\dws\doc\Sitraffic_smartCore _Manual_de.pdf
[4]	RILSA	Guidelines for Traffic Signals	
[5]	C-Control Javadoc		<installdir>\dws\doc\ccontrolclientlib-api\index.html
[6]	OCIT-O Documentation		http://www.ocit.org
[7]	C-Control.x interface definition		<installdir>\dws\doc\CControl.x
[8]	Sitraffic sX DWS Developer Guide		<installdir>\dws\doc\Sitraffic_sX_DWS_Developer_Guide_en.pdf

Preface

This paper shall give a quick overview of basic idea of the control model of the Sitraffic sX intersection controller.

A developer shall get a first idea of the dynamic behavior of the Sitraffic sX intersection controller, how to access information and control the Sitraffic sX via the C-Control interface.

This paper shall not substitute the C-Control specification itself (see [5]) but give a short introduction to it.

1. OMC Control Architecture

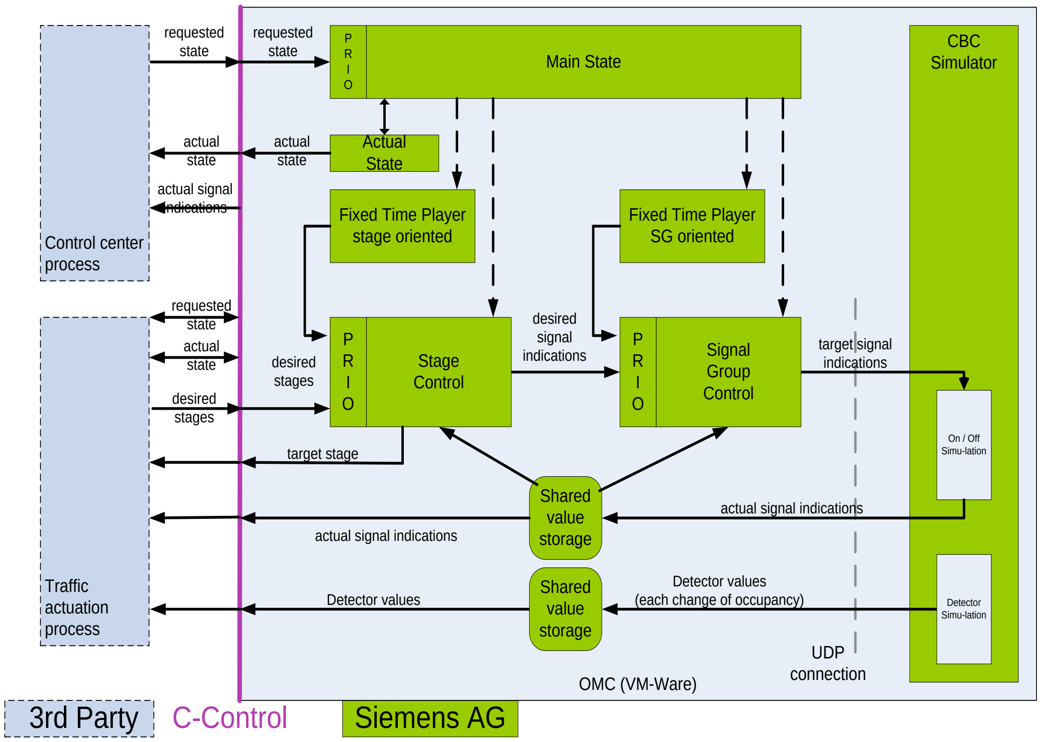


Figure 1: OMC Control Architecture

This diagram shows the main architectural components and main data flows of the OMC seen from the basic control point of view. Thus, it does not show all existing components within the OMC like configuration or Web interface.

It also shows the main idea of the C-Control interface.

This interface shall:

- Separate concerns of basic control, traffic actuation and control center connection
- Be IP network capable
- Widely independent from the programming language used

On the other hand, the diagram shows an additional simulation component (CBC simulation) that currently substitutes real hardware as far as needed.

2. Controlling the Main State

This chapter gives a rough overview how to control the main state of the Sitraffic sX intersection controller. The following picture shall illustrate the relationship of different sources of a switching request, control levels, priorities, the main state manager and different consumers of the switching requests.

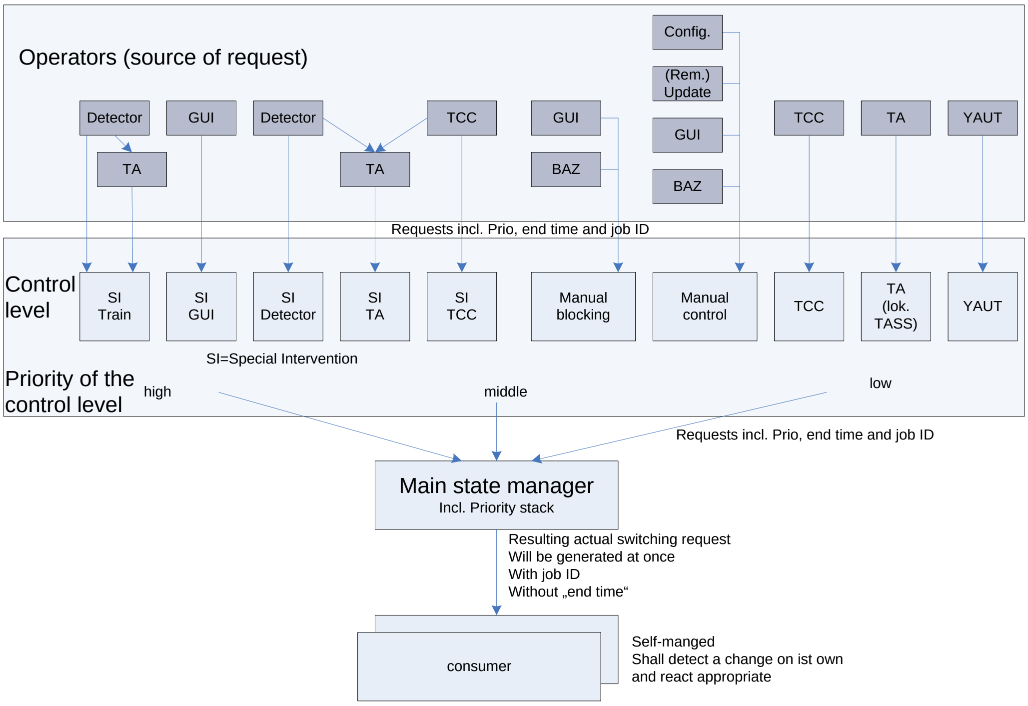


Figure 2: Basic control level concept

2.1. Control Levels

Different control levels of the intersection controller represent more or less the appropriate priority of a single control request. Priorities passed at C-Control must not exceed the range 0..999 otherwise an error will be reported.

Control level	Assigned priority	Example
Special intervention via detection of trains	150	A train is detected and a fast switch to a certain stage is enforced. Highest level of control.
Special intervention via GUI	140	Special interventions with lower priority are blocked. Highest level of manual control
Special intervention via detector	130	A detector takes control of switching events directly without the need of any traffic actuation.
Special intervention via TA	120	A high priority (local) traffic actuated control is working. Note: This control may NOT be superseded by the TCC (priority 30) for example
Special intervention via TCC	110	A traffic control center provides a special intervention. This may be a kind of centralized traffic actuated control
Manually blocked	50	Intersection controller is switched to “yellow flashing” because of road works. Unintentional control shall be avoided
Manually controlled	40	Manual local control via Web-GUI
TCC	30	The traffic control center is controlling the intersection controller from remote
(local) TA	20	A low priority traffic actuated control is working. Note: This control may be superseded by the TCC for example
(local) Time of day scheduler	10	(German: JAUT)
Default	-1	System default; Cannot be changed or deleted

Table 1 Control Levels and their Priority

Please note:

These priorities can be found within the java bindings also (see [5]; RequestedStatusPriorityCctrl).

2.2. Sources of Control Requests

A couple of sources may generate a control request.

- Detectors
- BAZ (hand panel)
- Sitraffic sX Service Gui
- Remote update
- Configuration (from local or remote)
- TCC
- TA

2.3. Relationship of Control Levels and Sources of Switching Requests

The source of the switching request has to assign an appropriate priority to the switching request according to Table 1 Control Levels and their Priority. The main state manager does not check the relationship of switching source and the priority intentionally.

2.4. Details of a Switching Request

2.4.1. Priority

Each control level has a fixed priority assigned. The higher the value of the priority the higher is its priority.

2.4.2. Time Range

Each switching request SHALL provide an “end time”.

The firmware always sets “now” for the “start time”. Each “switching source” is responsible to choose an “end time” that fits best to the appropriate switching request.

Basic considerations to choose the “end time”:

- It is recommended that a TCC already provides an “end time”
- A traffic actuation should consider two aspects: On the one hand it should be in near future. So if the TA has any kind of problem its switching request will time out and the Sitraffic sX firmware has full control again automatically (fallback concept). On the other hand, the end time should not be in the next seconds, in order to have enough time to send an update of the switching request. In most cases an “end time” 30 to 60 seconds after the current time will fulfill both requirements.
- If you want to switch “forever” use Long.MAX_VALUE / 4 for end time.

Please note: This “end time” may be changed at any time by means of a new switching request.

2.5. What Can be Controlled with a Switching Request ?

States as follows may be requested separately:

- Intersection state (none, on, off, ...)
- 4 partial intersection states (none, on, off, ...)
- Signal program
- Special intervention (see OCIT [6])
- TA on / off
- Individual traffic on / off
- Public traffic on /off
- 13 project specific modifications

Each of these switching requests has to be provided with Job-ID, priority and "end time".

Please note: It is a main part of the control concept that it is allowed to request only a part of these states at once.

Note: to run the intersection controller in a certain signal plan you have to control at least:

Intersection state = ON

At least one SubIntersectionState = ON

Signal program = configured signal program number

2.6. Special Control Situations

2.6.1. Concurrent Control with same Priority

If two clients take control with the same priority, the newer request will replace the older one despite of its content.

In particular the "end times" of the older request will not be honored.

2.6.2. Remove a Control Request

A single control request may be deleted by applying a new request with a duration of "0" ms. It is possible to delete all requests with the same priority with a single request.

2.7. Visualization of the “Priority Stack”

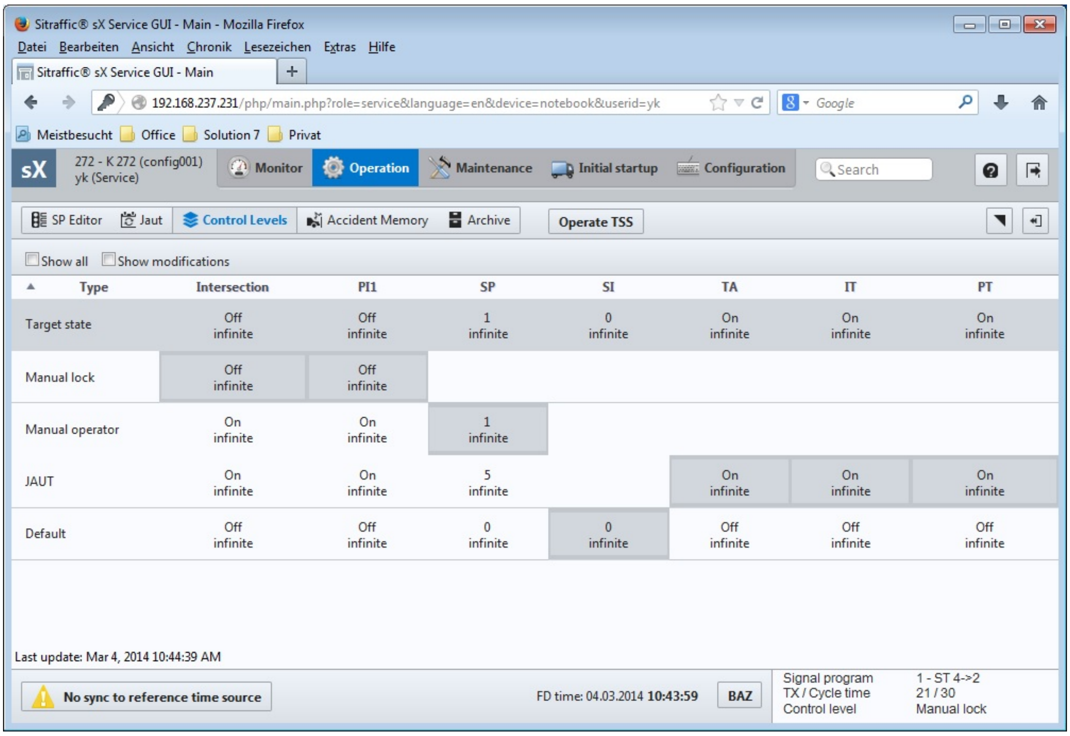


Figure 3: Sample priority stack

The sample above shows that three control levels (Jaut, Manual operator and Manual blocking) have different entries in the priority stack. The yellow line (Target state) shows the current result.

This result is computed for each column by “walking down” each column and “picking” the first cell not empty.

3. Stage Control

The stage control is the main component that can be influenced by a traffic actuated control in the current delivery.

In other words: the input of the stage control is the main output of a traffic actuated control process.

3.1. Input

Desired stages from different sources

Format:

Priority, desired stage, start tick, duration

See [5] CControlClient. addDesiredStageSwitchingEvents

- Start tick = -1 means „asap"
- An appropriate lag between invocation time and „start tick" is necessary.

3.1.1. Mechanism

In principle the mechanism of prioritization is equal to the mechanism described in chapter 2 Controlling the Main State. So a higher number means a higher priority. To make system behavior predictable only priorities as listed in Table 1 Control Levels and their Priority shall be used.

Note: more than one stage request may be given at a time by using a request list. This list may contain request with

- Different priorities
- Different start time and duration
- Mixture of both latter cases

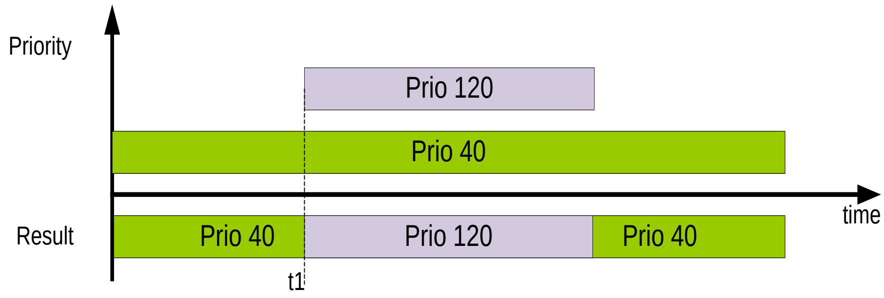


Figure 4: stage prioritization mechanism

This picture shows how two stage sequences with different priority (here 120=high and 40=low) are mixed to a resulting stage request.

3.1.2. How to Delete a Stage Control Request!

Add an ExternalStageSwitchingEvent with equal priority of the event to delete, start-time <= the start-time of the event to delete, a duration of 0, an arbitrary existent stage number. If you want to delete the grey stage at figure stage prioritization mechanism above, you have to pass an ExternalStageSwitchingEvent (priority=120, stageNumber=1, startTickTime=t1, validDuration=0)

If you pass an ExternalStageSwitchingEvent all stored stage switching events of the same priority with startTickTimes >= the newly passed are discarded.

3.1.3. Diagnoses

In difference to the priority stack of the “Main state” this priority stack cannot be seen in the users Web interface. For developer purposes, the input queue can be tracked using the JMX interface of the firmware process.

Input of queue:

http://<ip>:8002/ViewProperty/InputEventQueue/C10.SGControl.Phaser%3Aname%3DPhaser

3.2. Output

The output of the stage prioritization is a sequence of stages and stage transitions. A desired stage and its subsequent stage transition is called an “Atomic block” because both shall not be separated.

In the next working step each atomic block is converted into the data stream called “Desired signal indications”.

In the JMX GUI this queue can be debugged with this link.

http://<ip>:8002/ViewProperty/StagesToPlayOut//C10.SGControl.Phaser%3Aname%3DPhaser

Additionally an information called “Target stage” is computed, that fits with the “Desired signal indications”.

Note: This “Target stage” does not represent an “actual stage”, because the mechanism described below may alter or drop the generated “Desired signal indications”.

4. Signal group control

This module is the smartCore module to process a signal plan. If only a signal group oriented signal plan shall be processed, this module can operate without any input from the stage control.

4.1. Input

Desired signal indications from different sources

Format:

Signal group, start tick, duration, indication state of signal group (green state, red state)

4.2. Prioritization

The mechanism is the same as described above in the chapter “stage control”.

Note: the prioritization here has to be done for each signal group.

4.3. Request Correction

The prioritized signal indications might violate minimum time or inter green time constraints. A module within the signal group control tries to correct the faulty requests if possible. Note: if this correction mechanism is activated due to a faulty signal plan, the “target signal indication” will be different from the signal indications within the original signal plan.

4.4. Output

Target signal indication

Format:

Signal group number, color, start tick time

A convenient way to see this output is to look at the Visu STP (Sitraffic sX Service Gui).

5. CBC Simulation

Main task of the CBC simulation is to let the firmware of the OMC operate as if a real CBC is connected, which is the signal monitoring unit of the Sitraffic sX controller. The CBC Simulation does not reimplement all features of a real CBC but the most important ones. Those main features are listed here.

5.1. Generating “devTicks”

Why using “devTicks” ?

Most of the basic control software on the OMC intentionally does not use the system time or equivalent time sources to measure the time elapsed.

Time steps caused by synchronizing the system clock with any external time source would cause unpredictable results.

Therefore, the OMC firmware expects to be “clocked” by an external time source with strictly monotonic behavior. This time source is called “devTicks”. Each “devTick” last 100ms. These “devTicks” are generated within the CBC.

Thus, the CBC simulation is able to generate “devTick” telegrams each 100ms starting with “0” after a restart of the simulation.

With every telegram sent the devTick counter is increased by one.

5.2. Reflecting Target Signal Indication

The CBC has to ensure that the target signal indication is visualized on the real lamps outside. If no signal monitoring error occurs this “target signal indication” equals to the “actual signal indication”.

The CBC simulation assumes an error free operation mode and is reflecting the target signal indication” as “actual signal indication”.

Note: If the intersection controller is switched off, the CBC simulation is sending a “dark” as actual signal indication.

5.3. Reflecting Intersection States

Quite similar to the target signal indication the CBC simulation is reflecting the intersection- and partial intersection state.

5.4. Simulating Detector Values

Currently two different ways exist to simulate detector values.

- A kind of autonomous wave form generator
- A remote controllable impulse generator

The first kind of simulation is controlled via a simple XML file. You can configure a frequency and an occupancy rate for each detector. With startup of the CBC simulation this configuration file is read and the appropriate "waveforms" are generated and sent to the OMC.

The latter mechanism uses the JMX interface to inject a single burst of occupancy. This mechanism may be used manually via the JMX http interface or automated via the JMX RMI interface.

Details of the latter mechanism are described in the "Developer Guide" (see [8]).

6. Dynamics of the C-Control Interface

6.1. Technology of the C-Control Interface

The standardized part of the C-control Interface is on the network layer (C-Control is not an API).

The C-Control interface is implemented on top of ONC-RPC, also known as sun-rpc. See RFC 5531.

Interface specification see [7] CControl.x.

The Java bindings generated using the interface definition file CControl.x are included in C-Control client library CControlClientLib.jar. The generated java bindings and the provided helper classes are NOT the interface specification.

Please note:

Some use cases refer to other technologies like JMX. This technologies are for testing purposes only and are not part of the C-Control interface.

In other words: Interfaces not described in "CControl.x" will most likely not be available in a productive environment.

6.2. Subscription to RPC Service

A C-Control client may subscribe for spontaneous value delivery of:

- detStates
- actualSignalIndications
- actualStatus
- requestedStatus
- targetStage

by creating a CControl_CcontrolDataReceiver_ServerStub an then calling the CControl_CcontrolService_CltIF.subscribeXXX methods (see [5]). For JAVA clients we recommend to use the DataReceiver in conjunction with DataReceiverStarter provided with CControlClientLib.jar.

Each subscription is answered by an initial update.

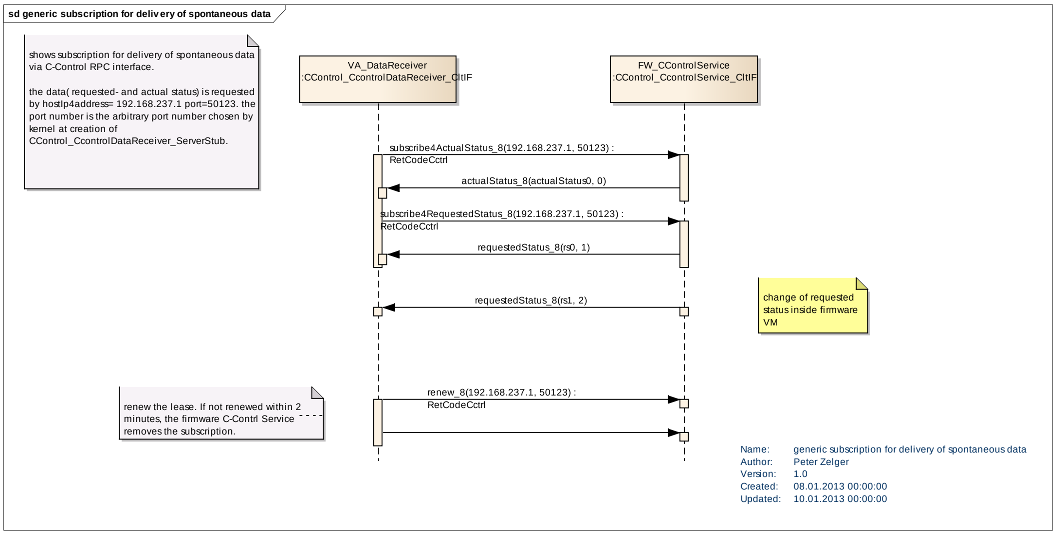


Figure 5 subscription to the RPC service

6.3. Value Delivery from Firmware to C-Control Client

The C-Control interface guarantees a dev tick time monotone value delivery only within the same value type (detStates, actualSignalIndications, actualStatus, requestedStatus, targetStage).

There is no fixed order of value type guaranteed. E. g. you cannot assume that at reception of detStates the signalIndications of the same tick are already received.

The detStates and actualSignalIndications are delivered each 100ms at average. A C-Control client has to deal with a jitter (typically less than 1 second). The other values (actualStatus, requestedStatus, targetStage) are delivered on change.

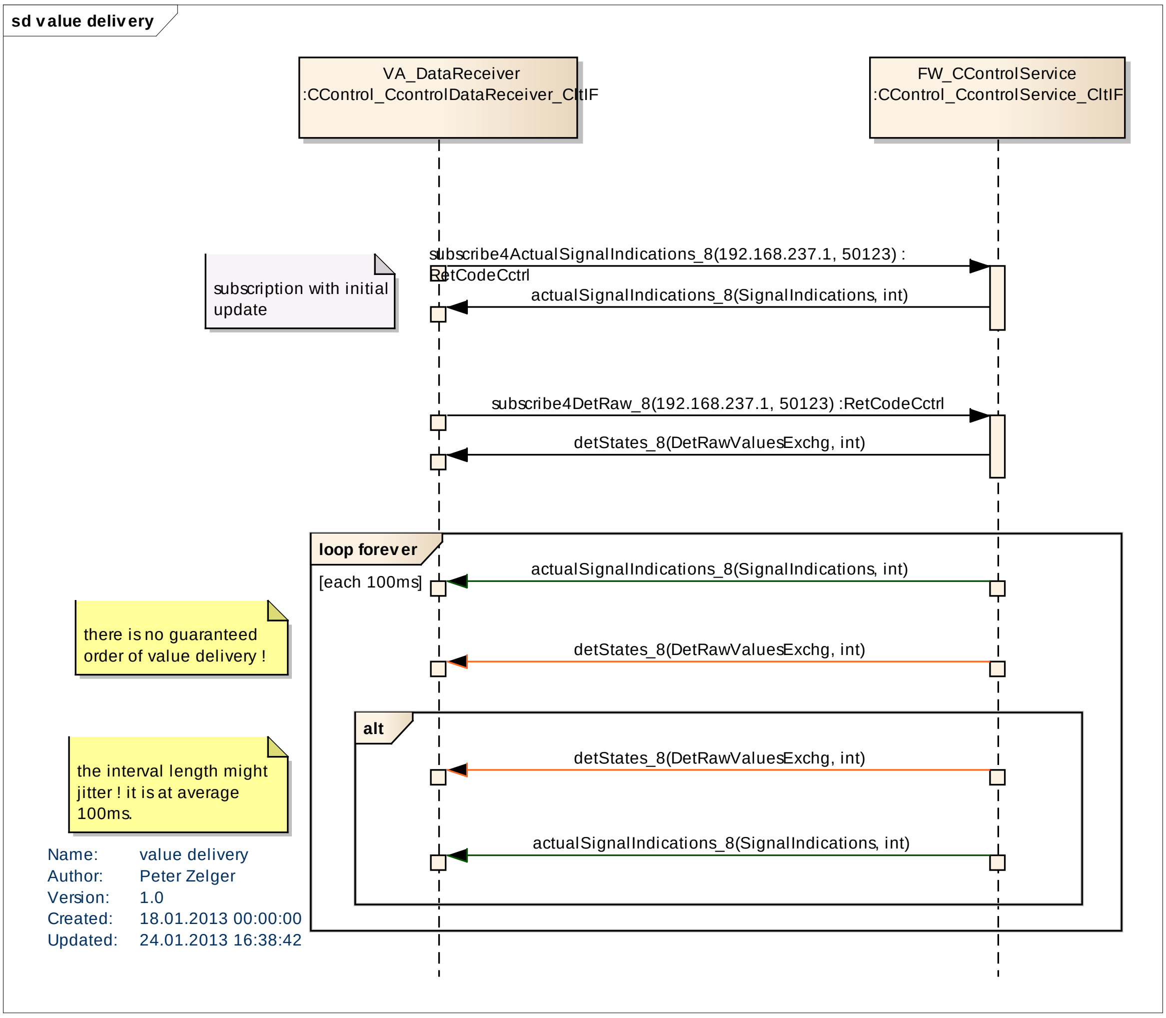


Figure 6 Constrains of Value Delivery

6.3.1. Dynamics of Detector Values Delivery

The initial update contains the error states and occupancy value of all detectors.

After the initial update the CControl_CcontrolDataReceiver_CltIF. detStates_8() gets called typically each 100ms, reporting only changes at the detector occupancy value or error state see[5].

6.4. Maintaining actual Status of Traffic Actuation

A traffic actuated control SHALL maintain the actual status of the modification:

- TOM_ModTD
- TOM_ModPublicTransport
- TOM_ModTDIndividualTraffic

(see [5] ActualStatusCctrl#modifications)

To do so it is recommended that the TA looks into the requested status to obtain the desired state of traffic actuation.

The activity chart below shows the handling of the macroscopic states by traffic actuation control process.

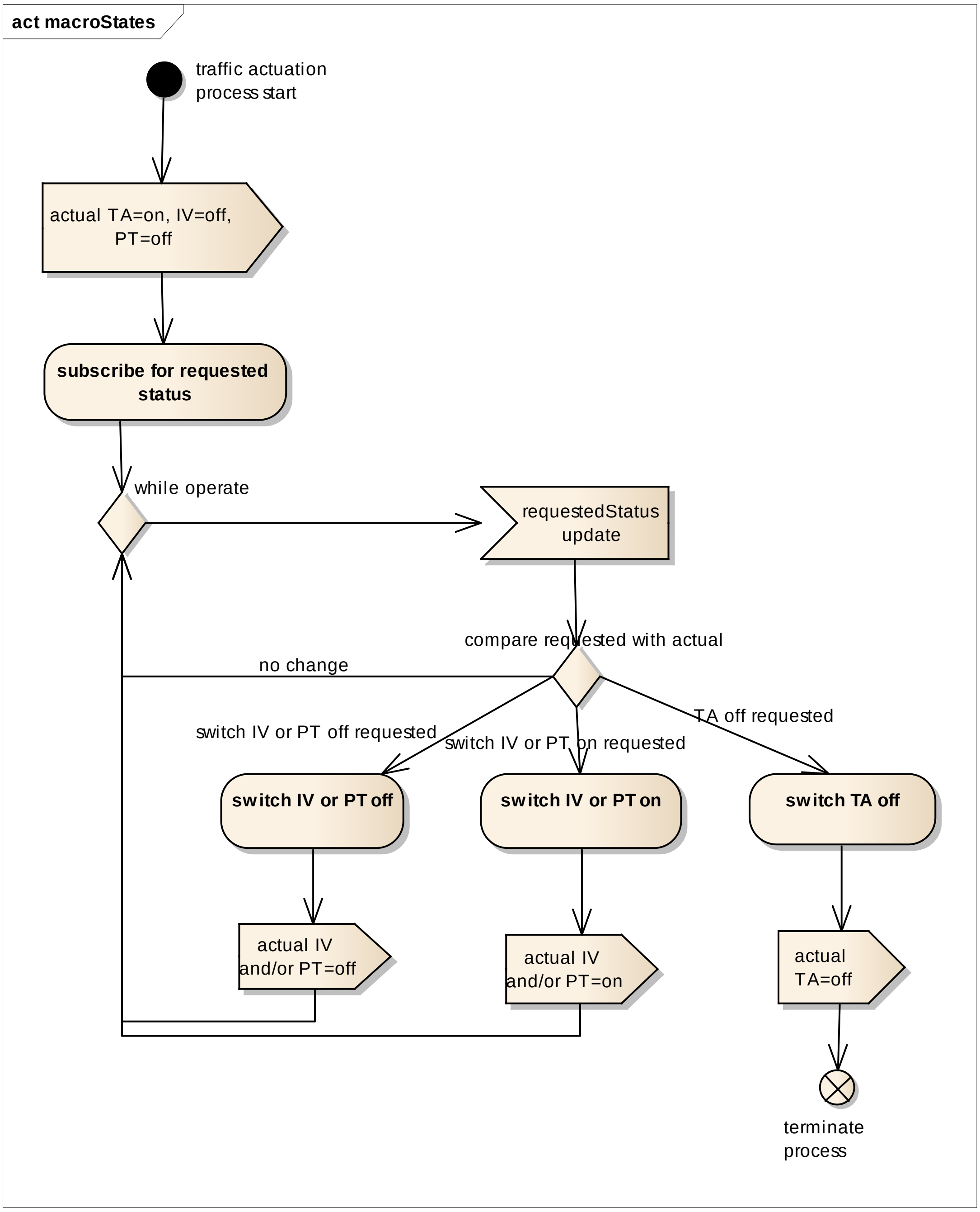


Figure 7 Maintaining actual Status of Traffic Actuation

7. Index

Further information
is provided by:

Siemens AG
Infrastructure and Cities Sector
Mobility and Logistics Division
Road and City Mobility

Otto - Hahn - Ring 6
D-81739 München

The information in this manual contains descriptions and features which can change due to the development of products. The desired features are only binding if they were agreed upon conclusion of the contract.

Order no. A24730-A890-B002

www.siemens.de/traffic