Training module

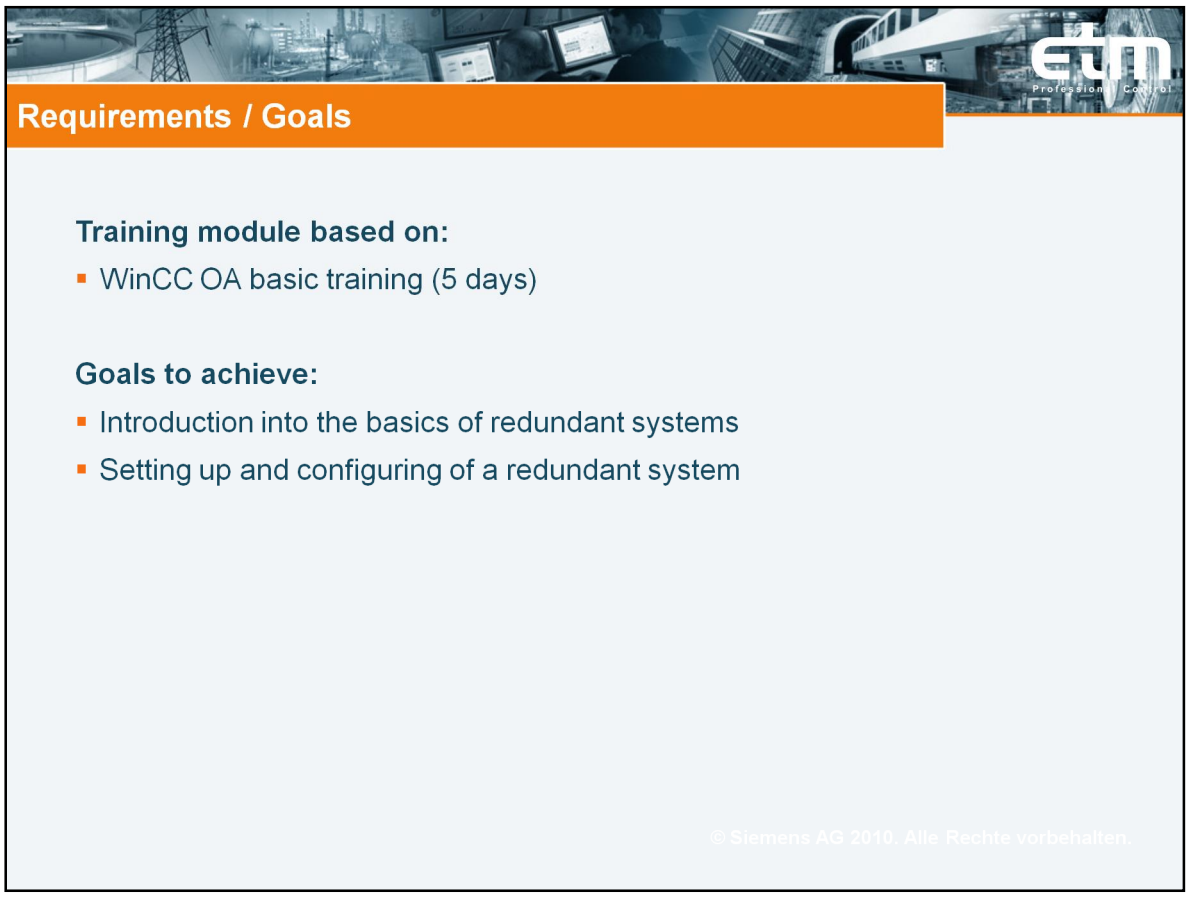
WinCC OA
Redundant Systems

Slides

Exercises

Comments

Content



Definition:

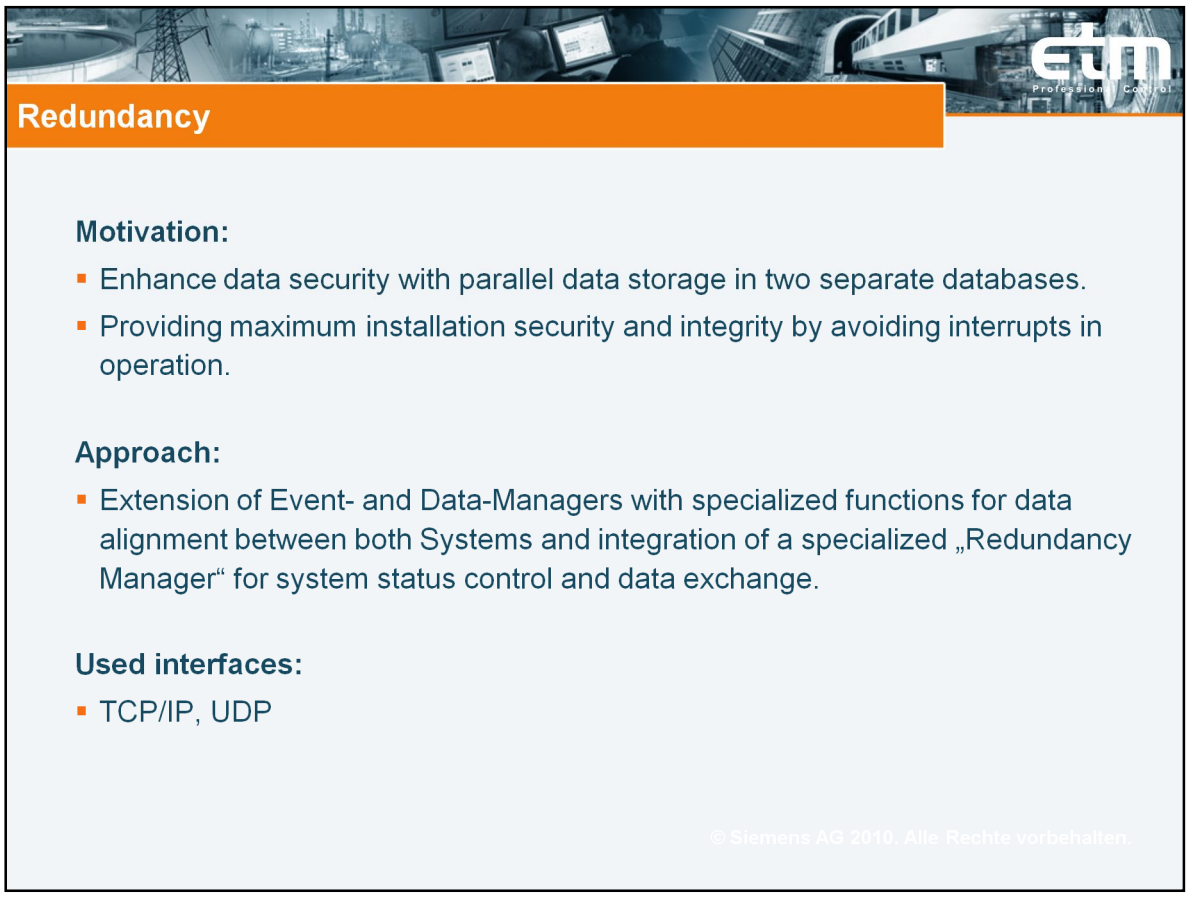
A system is considered **redundant** when there is a greater number of functional blocks or technical means in one single unit available than it would be really necessary to maintain normal operation.

Some different kinds of redundancy:

- **Hot standby redundancy**: the additional technical means are continuously in use and are stressed exactly in the same way as the primary unit.
- **Warm standby redundancy**: the secondary unit will be carrying only a small workload which will result in a reduced stress until the primary unit fails and then take over the process completely.
- **Cold standby redundancy**: the secondary unit is at a real “stand by” state and therefore experiencing no load at all untill the primary system fails. However usually it takes some time till a cold standby system is coming online in case of an emergency (this will result in a loss of data).

High availability and reliability are becoming more important in the automation technology. Even a short breakdown can result in heavy costs and security risks. The redundancy is used to guarantee plant management without any adverse effects in case of a failure and thus loss of data and associated problems do not occur.

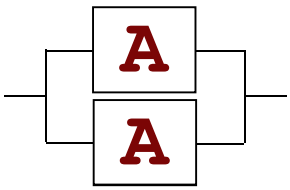
The reliability in a redundant system with **WinCC OA** is implemented via **hot standby**. Hot standby is a hardware independent solution for high availability. This is a security concept consisting of two server systems connected with each other. Both servers are operating permanently and are subject to same functional demands (only one server is, however, always active. The second server synchronizes the data at runtime with the primary unit). If a unit fails a "flying switch" is executed and the server, which was passive until then, takes over the control. Thus, the access to data or functions is always guaranteed.



Note:

Availability

process A has an calculated or measured availability of **0,90**

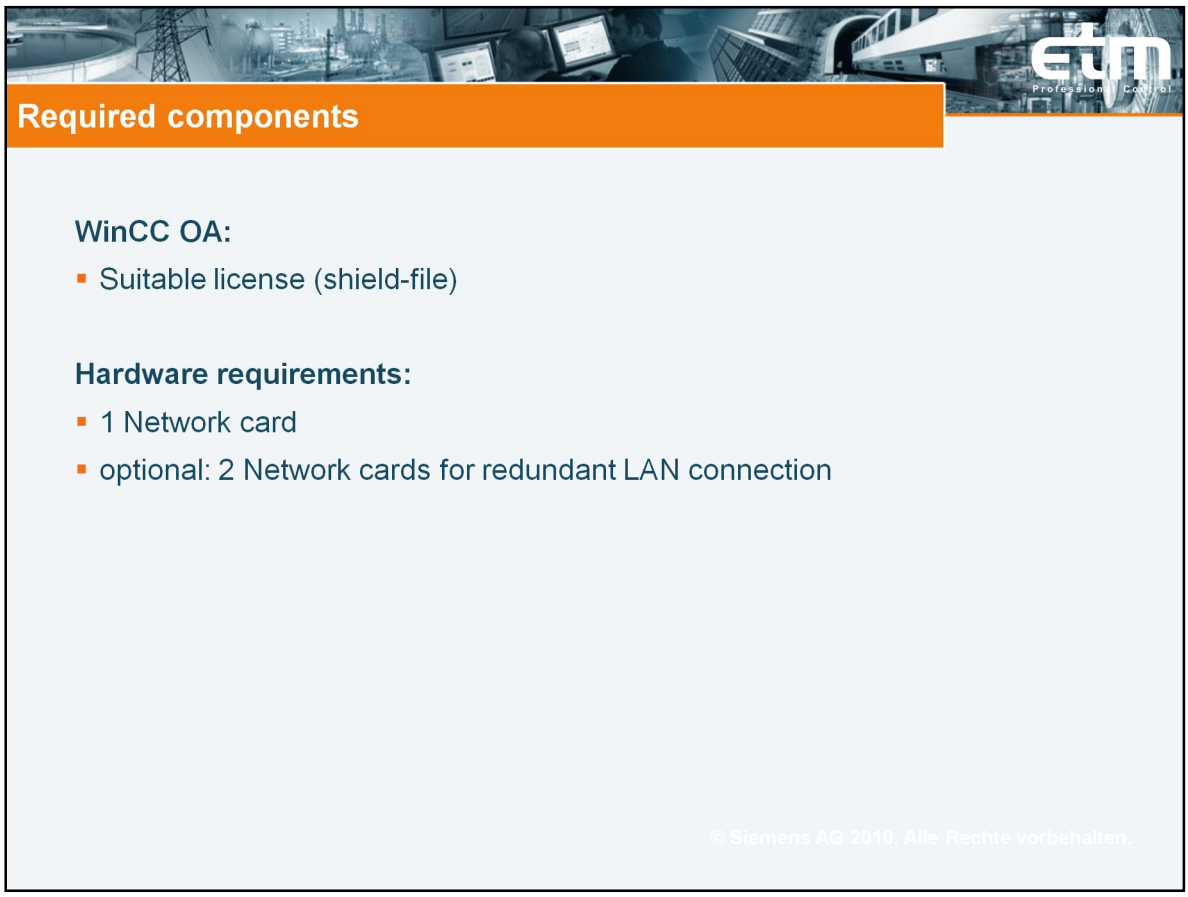


in theory, two parallel processes of the same kind (A) will than have a calculated availability of A v A=1-(1-0,9).(1-0,9) = **0,99**

But the real life shows, that the mechanism of redundancy on its own will introduce an additional source of failures. Therefore the above example may reach not 0,99 availability but maybe only from 0,96 to 0,98.

Attention:

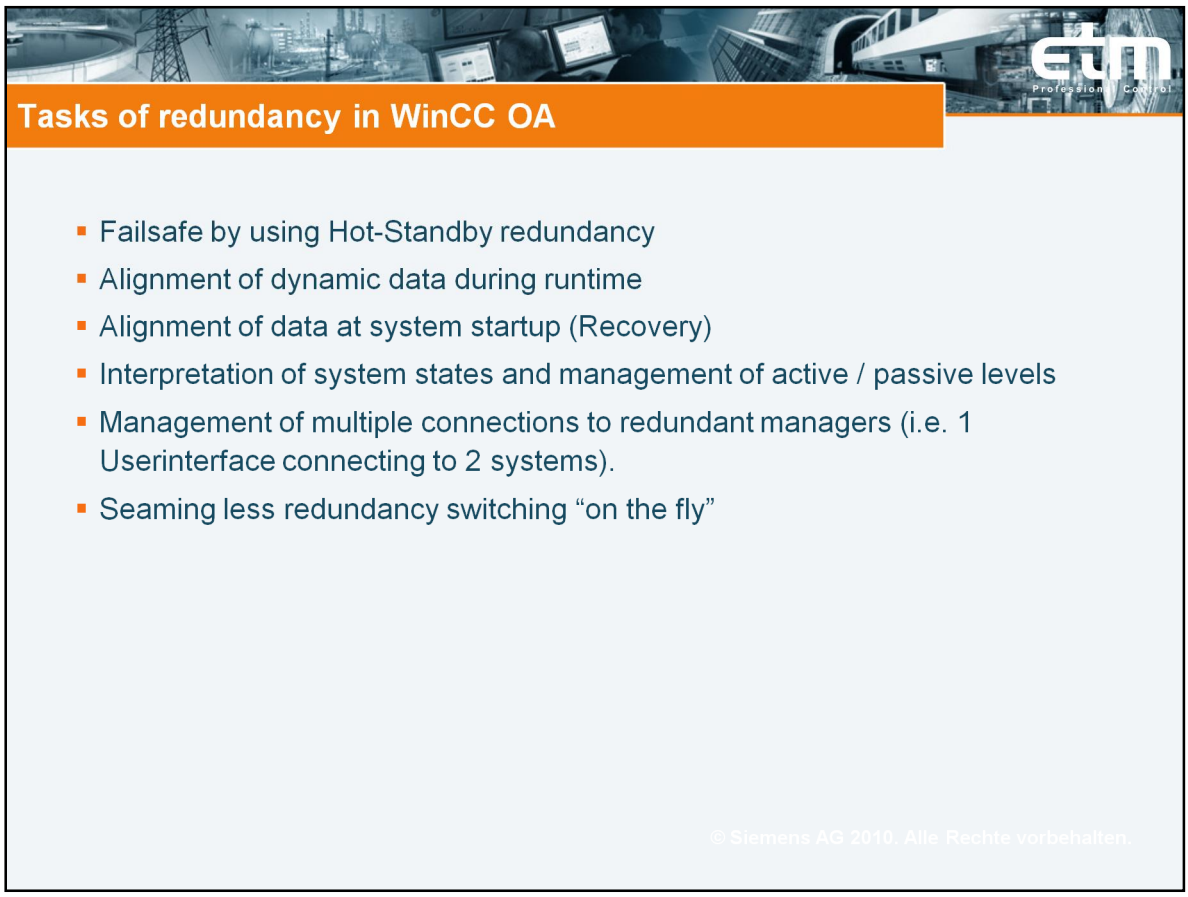
The above mentioned numbers for the system availability is only meant as an example and does not reflect the availability of a real WinCC OA system (the real availability numbers may vary in a wide range, depending on the quality of the code, a user adds to his project (panels and scripts) and of course also the quality of some user created drivers or managers using the WinCC OA API interface).



Note:

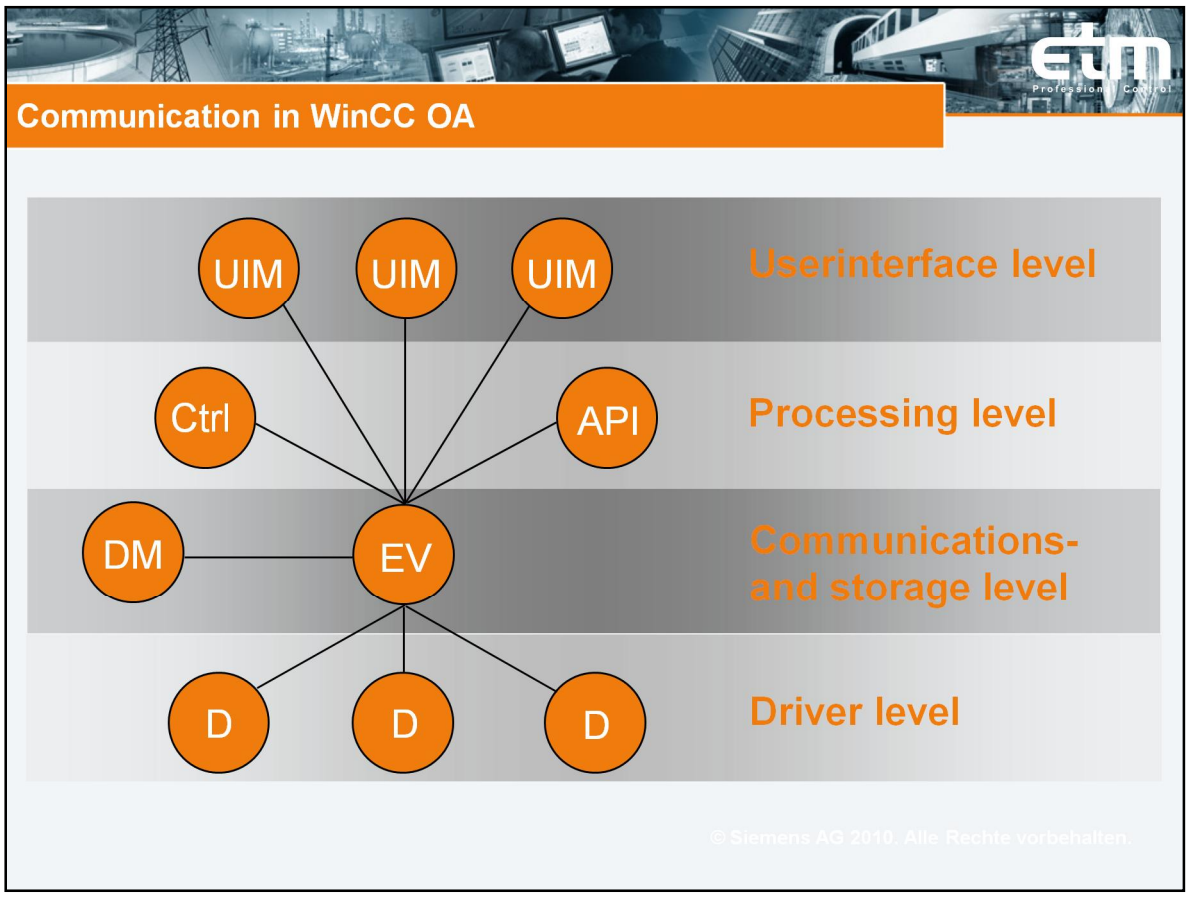
For a redundant WinCC OA system at least two hosts with identical operating system and identical WinCC OA Version installed to it are needed.

In addition to that the project-path has to be the same at both servers (make sure that there is the same number of hard disk drives installed to both servers).



Tasks of redundancy in WinCC OA

- Failsafe by using Hot-Standby redundancy
- Alignment of dynamic data during runtime
- Alignment of data at system startup (Recovery)
- Interpretation of system states and management of active / passive levels
- Management of multiple connections to redundant managers (i.e. 1 Userinterface connecting to 2 systems).
- Seaming less redundancy switching “on the fly”



Note:

Client-Server - concept

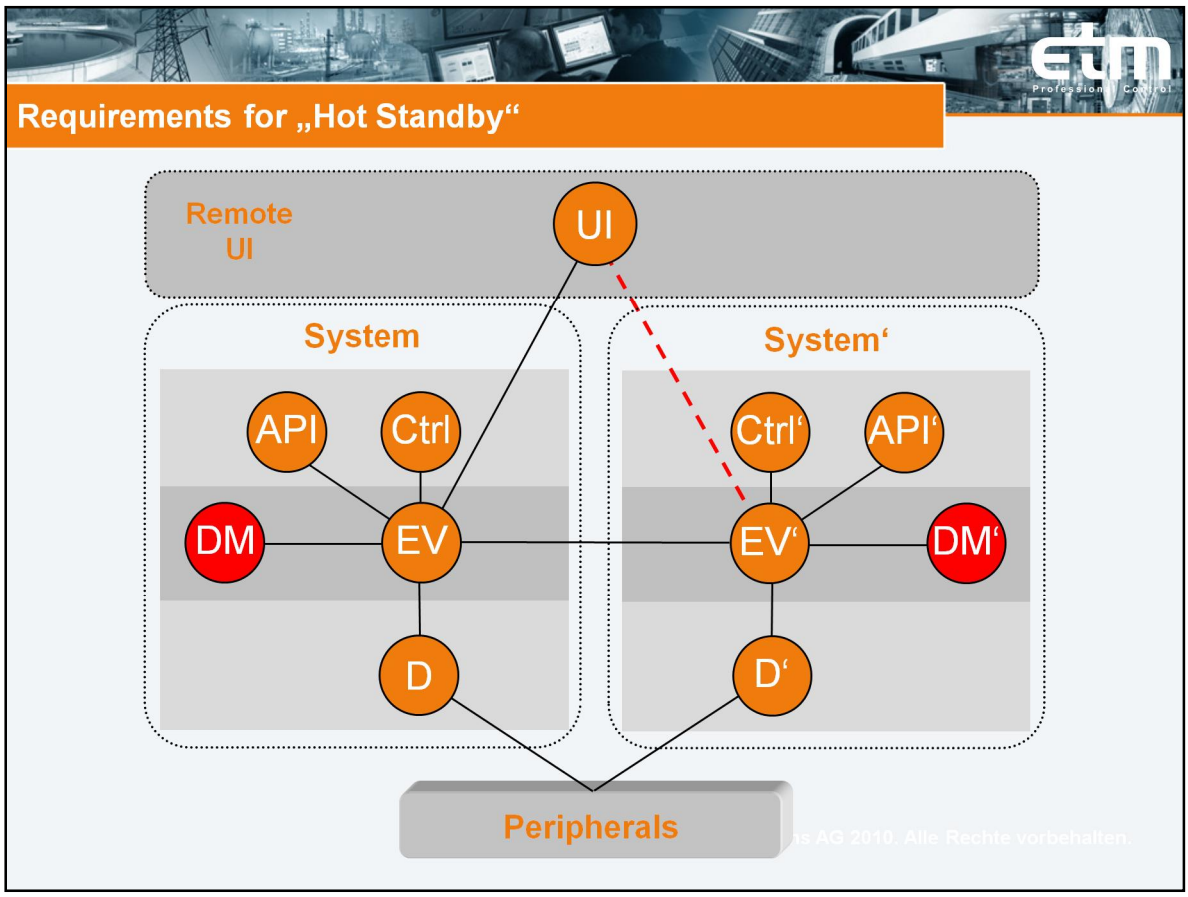
The Event-manager (EV) and the Data-manager (DM) are acting like a server; the other managers are acting like clients. The connections between them is done throughout TCP/IP.

The communication amongst the manager is done by the use of proprietary “messages”. The structure of this messages is defined by the API, the handshake mechanism is defined there as well.

The different managers (EV, D, DM) will care about “their” configs (i.e. the parameterization of the periphery addresses and also some calculations in respect to these values are directly carried out by the certain driver and not by the Event manager).

The EV has complete knowledge of all process states (i.e. all last recent values with their actual timestamp and statusbits, from every process variable).

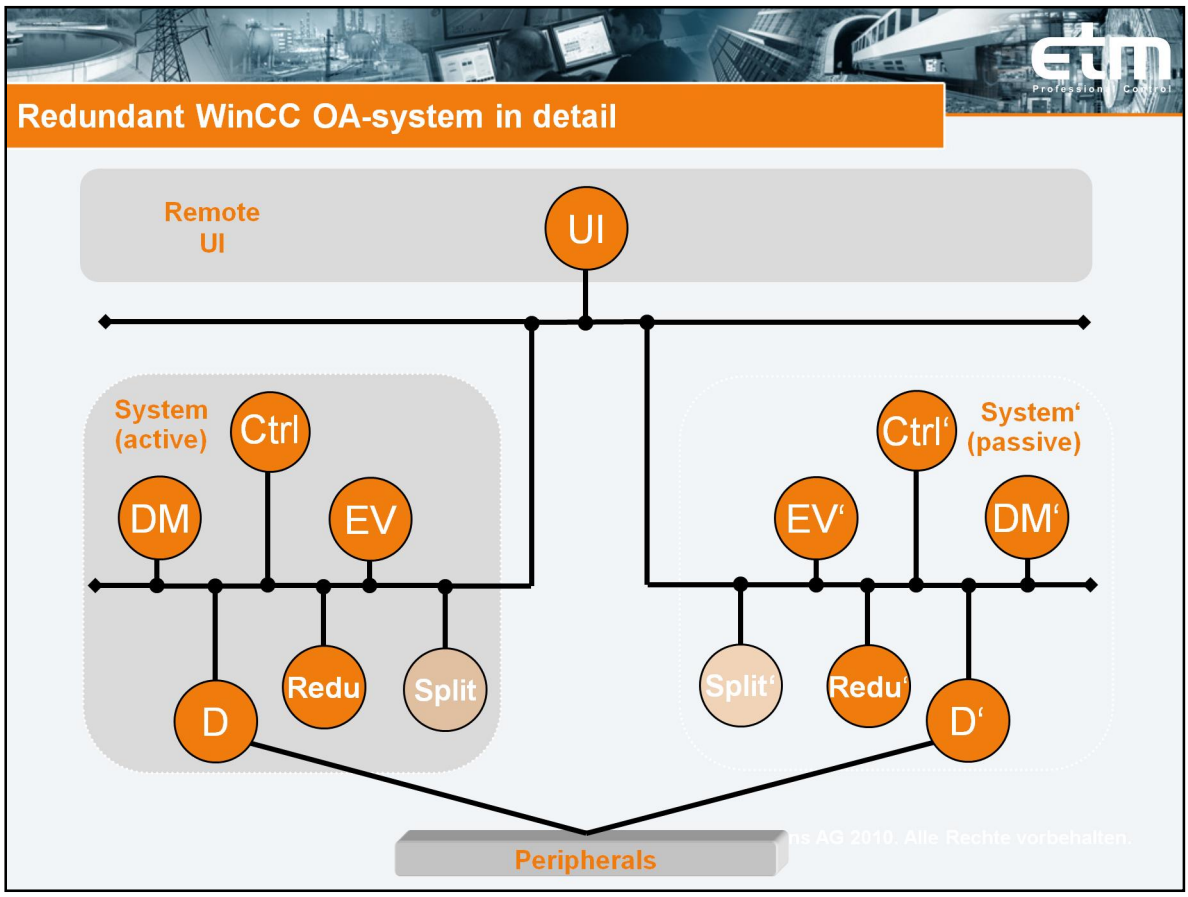
When some data is queried by any other of the connected manager, the **EV** will answer this messages if the query regards **actual** data, when **historical data** is queried, the message will be forwarded to the **DM** and he will return the data via the EV.



Note:

The requirements to a complete and comprehensive „Hot-Standby-Redundancy“ are as follows:

- Make sure that the process data is stored in a separate and redundant way.
- The communication between the managers of the two systems has to be ensured in a reliable way.
- Make sure that the peripherals are getting commands only from the active system
- Definition of exact rules and precise procedures in the case of a connection loss between the two redundant systems.
- Eventually keep a few UI workstations ready to use if one important UI is failing. For normal UI workstations are not built up redundant but used as remote workstations that connect to a redundant server pair.



Note:

The basic concepts of redundancy in WinCC OA:

Redundancy is a manager internal feature, that all WinCC OA managers are supporting inherently.

When using WinCC OA as a redundant system, than all the required managers that are part of the project will run on two separate, but connected systems. The remote UIs are in most cases installed on non redundant workstations (but for normal there are several equal UIs at a control center) and they are all connect to the central redundant server pair.

The UI holds connection to both EV-managers, but only the active one takes the messages sent from the UI to update its process view and will also forward this value changes to its passive partner.

The passive EV-manager on his turn also updates its own process view, therefore the two separate process views will every time stay in sync.

The dynamical switching from active to passive and the other way round on the other system may therefore take place very fast and is also carried out by the EV-manager.

For the initiation and synchronization of a redu-switch the EV-manager can rely on a specialized Redu-manager, a special control script and several internal data points.

The connected UIs will follow automatically to the current active server and therefore it is ensured that they will show the correct and actual process view at all the time.

(From the view of an UI, The redundancy switch will take place completely in the background and therefore it will be seaming less with no interruption to the operability of the UI.)

Redundancy state:

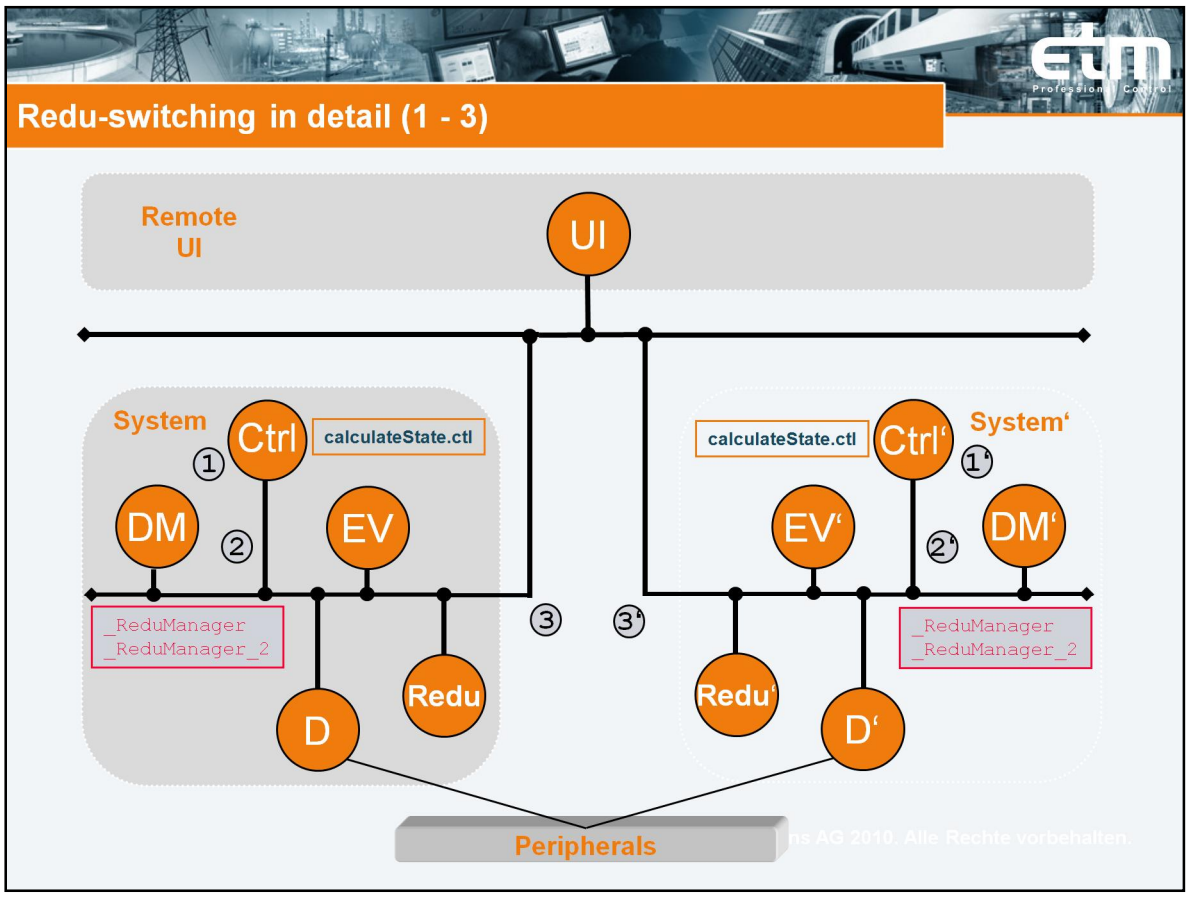
The redundancy state will be monitored and controlled by the Redu-manager, because they are monitoring all the vital information (like error-states, redu-states or connection-alive-states) not only from its own system but also from the redu-partners system. Those states are all together used to calculate a system-error-state, that are then compared to each other. The system with the better (lower) error state will become the active system at any time. These calculations are carried out by a control script (`calculateState.ctl`)

Error states:

For the calculation of the error-states, the states of all the existing WinCC OA managers and their current connection as well is considered. Also values from some data points or the current memory consumption may be monitored to contribute to the error states.

Because all this information is stored on data points in the internal WinCC OA database, an individual change to the parameters of this calculation is easily done.

All these changes may be incorporated by using the system overview panel with no need of changing some internal scripts or doing some advanced programming, etc.

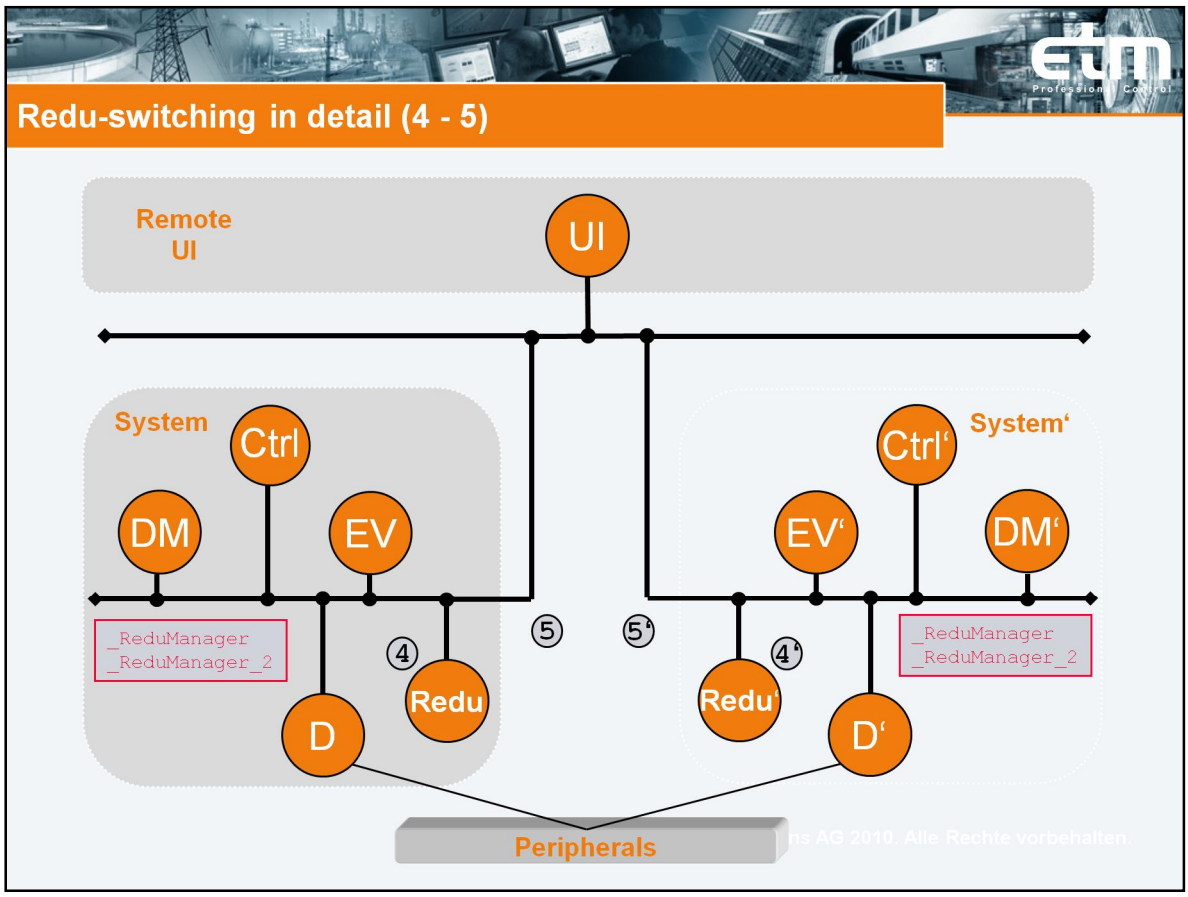


Note:

The UI is connected to both systems (EV), but will only accept messages with updates to the process view, from the currently active system (in our example here, the left system is the currently active one). Because the UI is connected to both systems; commands will also be sent to both of the systems EV-manager, but nevertheless the passive EV will drop this message and therefore ignore it

The Redu-managers will establish a direct connection between each other, witch is used in case of a redundancy switch for synchronization reasons.

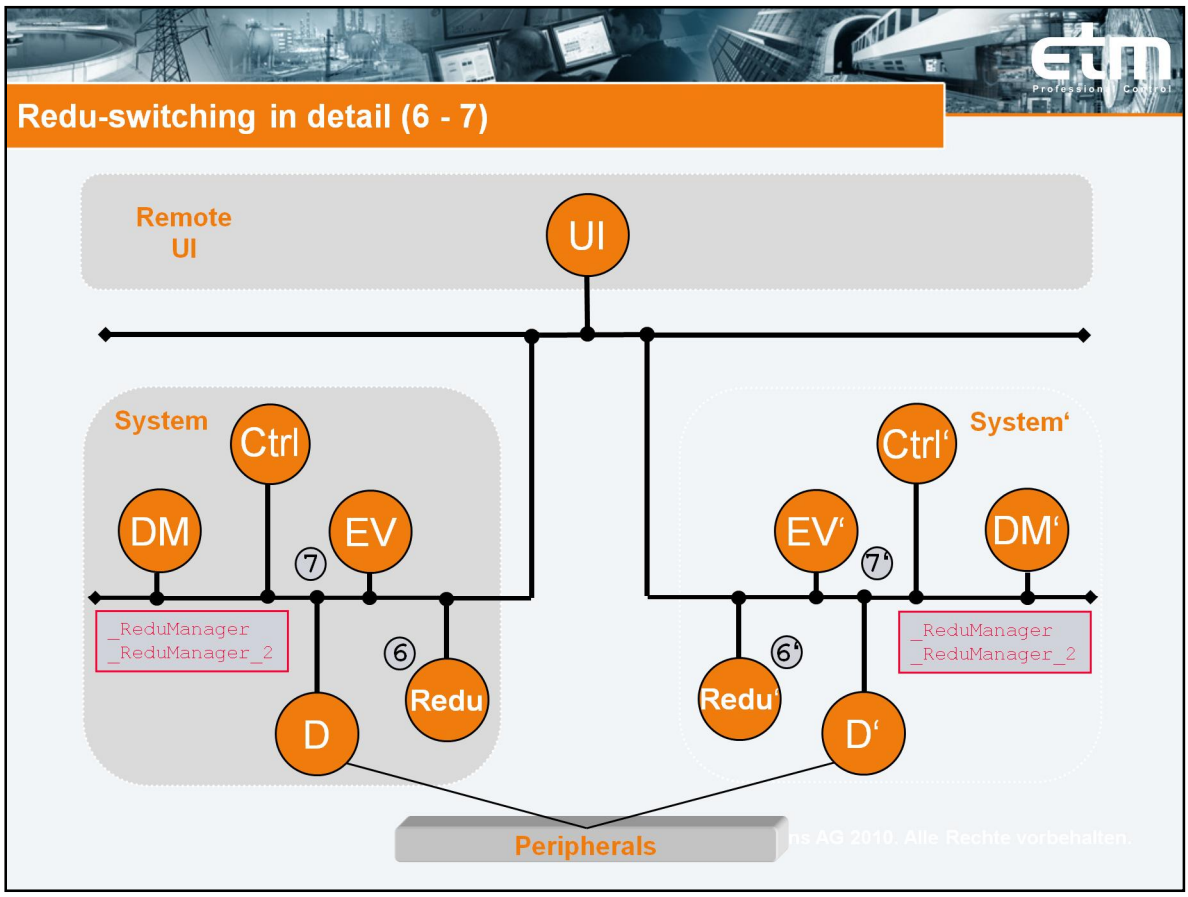
1. Both `calculateState` scripts will calculate the current error state for each system (the details were mentioned on the slides before).
2. The results are stored on data point elements of the internal data point named _ReduManager (or _ReduManager_2 for the second system)…
3. … and by the means of a fwdDP command (placed in the config.redu-file) the calculated values from the own system will be forwarded to the other systems _ReduManager (or _ReduManager_2) data points.



Note:

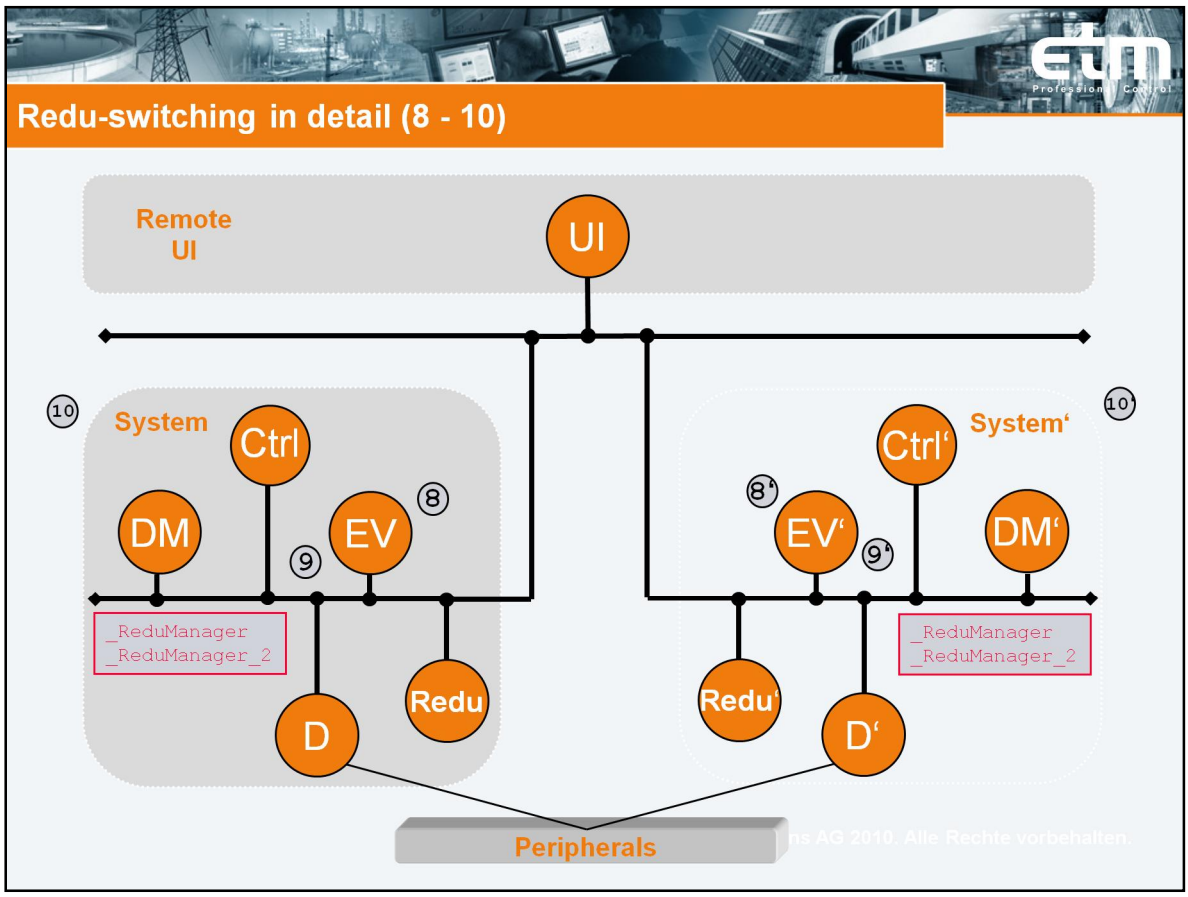
4. Each Redu manager now knows about the error states of both systems and under consideration of some additional rules they may derive now the actual redundancy state.

5. In the case that the redu managers have to invoke a redundancy switch of their systems, they will exchange this information about the planned redu-switch to each other (for double checking reason).



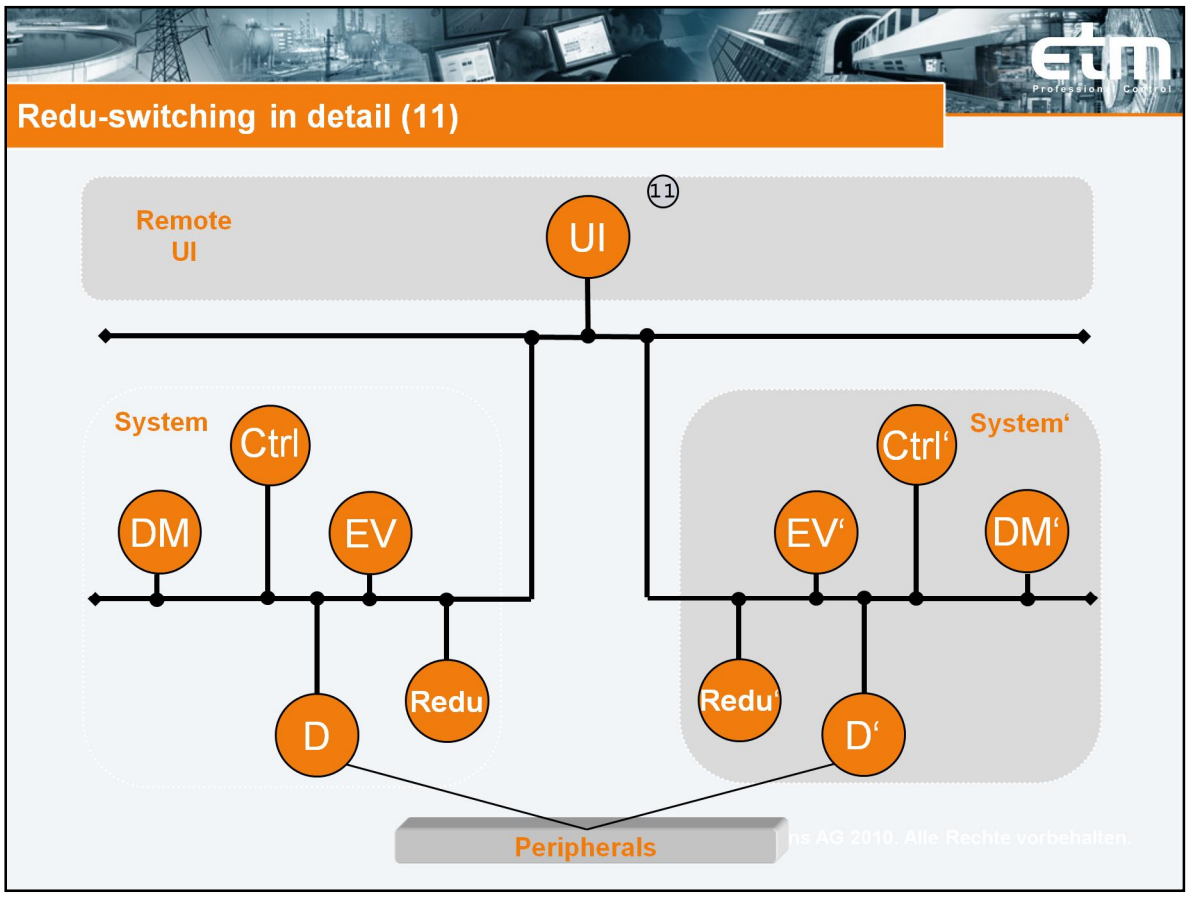
Note:

6. If both redu manager came to the same conclusion, that there is a redundancy switchover necessary (this should be the fact in the very most cases), the local EV-manager will be informed by its redu manager to carry out the redu-switchover.
7. Both EV-manager are setting their `_ReduManager`- data point to the appropriate values, that will be for instance the current and upcoming redu state. This change of data will now trigger the redu-switching..



Note:

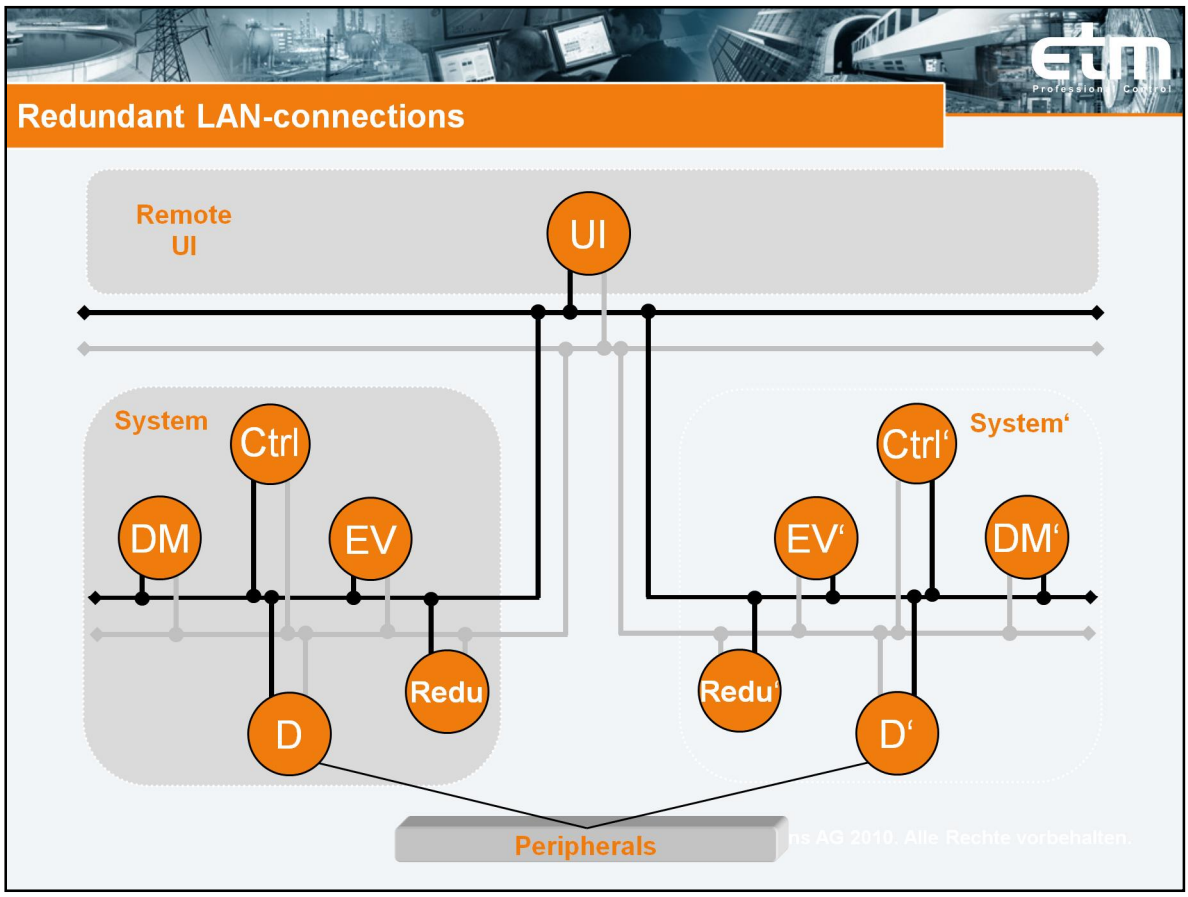
8. Both EV-managers send out a system message that informs the other EV-manager (and also any other manager in the system) that its state is going to change now from active to passive or vice versa. This message has to be acknowledged by the other EV-manager per handshake-mechanism …
9. … after that the now changed redu state is written to the internal `_ReduManager` data points…
10. … at this stage the redu-switching is finished.



Note:

11. The UI is also receiving the system message from the so far active EV-manager and switches automatically to the other, from now on active system.

The whole switching procedure will take place unnoticed from the user, completely carried out in the background (of course by putting on the system overview panel the actual active or passive states and also the redundancy switching may be observed by the user).



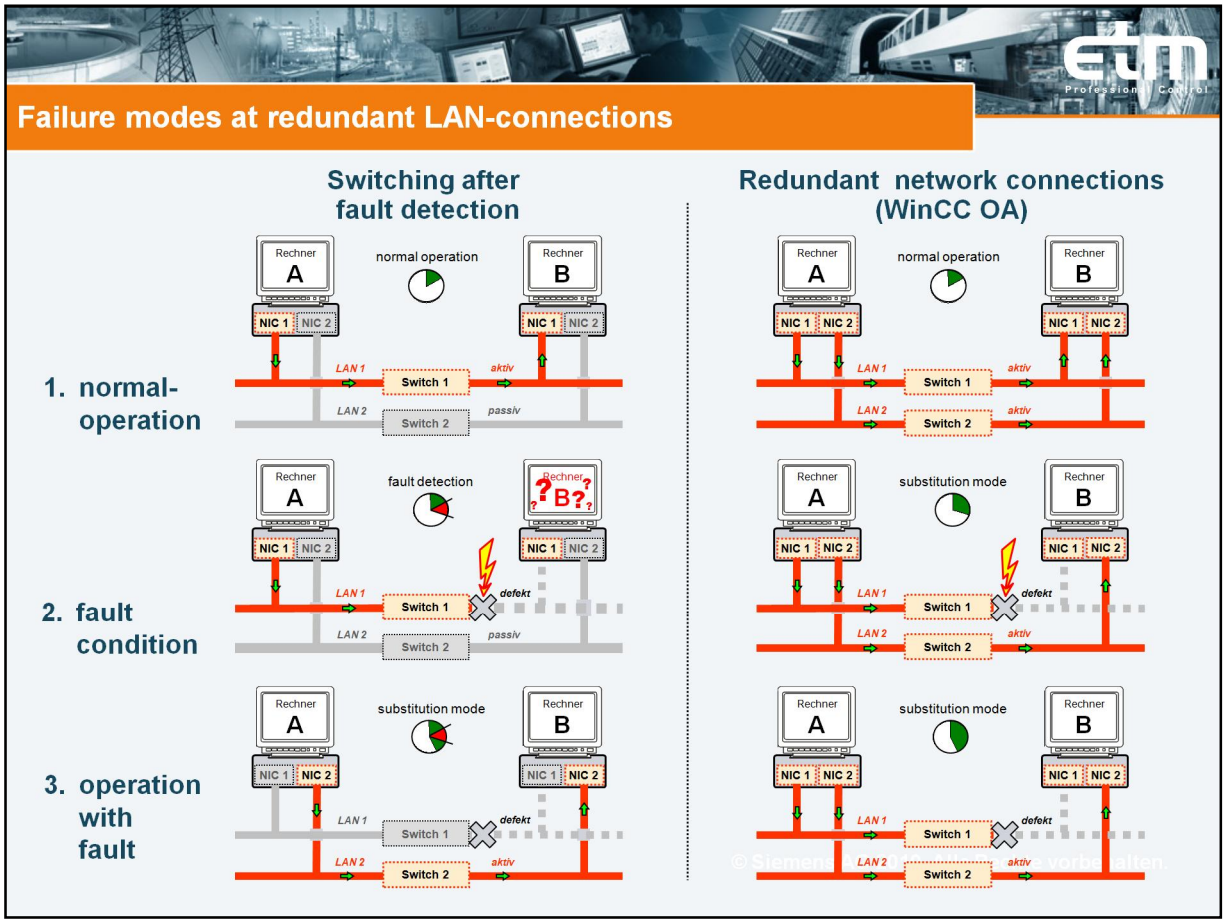
Hint:

In order to achieve a better fault tolerance WinCC OA managers may also be connect to other WinCC OA managers running on different computers via redundant network connections (this means a connection established by using two physically separated network connections, i.e. two network cards at each machine).

If you configure redundant network connections in a (distributed) system, the messages are sent using two physically different connections. When the system receives a message, it accepts the message immediately. If the system then receives the identical message through the second connection, it discards the message. (i.e. the system will not delay the execution of the first incoming message only to wait for the receiving of the message through the second network connection).

This can be done because every message in WinCC OA will have a unique, constantly rising message id. Messages with lower ids than the last recent one received can be discarded safely.

Is one of the networks failing there is no need for doing a network switch or something like that, because the messages keep coming to the other system at least once.

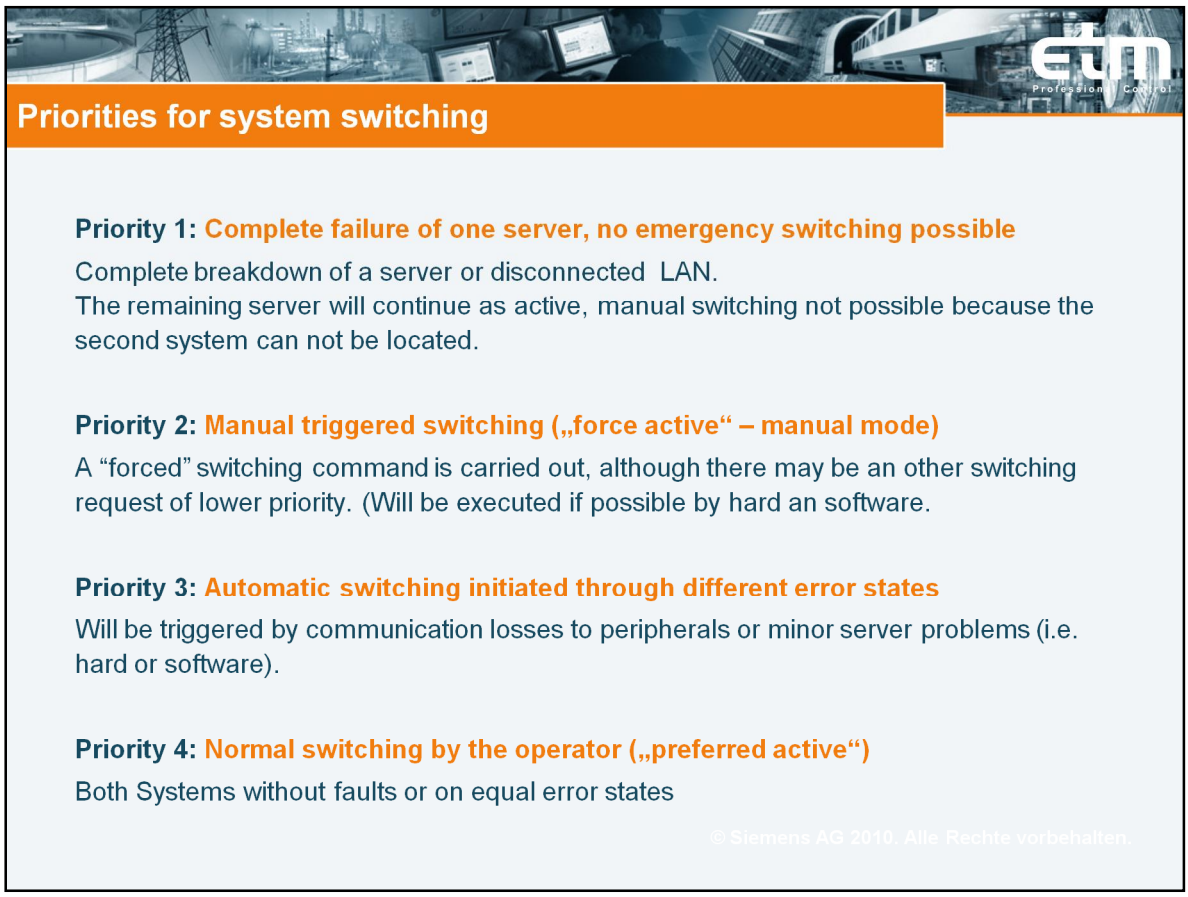


LAN	**Local Area Network (PC Netzwerk / Ethernet)**
NIC	**Network Interface Card (Netzwerkkarte)**

Hint:

When using a redundant LAN connection the IP-forwarding has to be disabled at the operating system level. Otherwise the OS would try to reroute the messages from the broken connection to the still working connection, which will double the network traffic over this connection and therefore maybe lead to collisions and packet losses.

Also the detection of the network failure would be a little bit tricky, because in the case the OS found an alternate route the “broken” network connection is also shown as up and running.



Note:

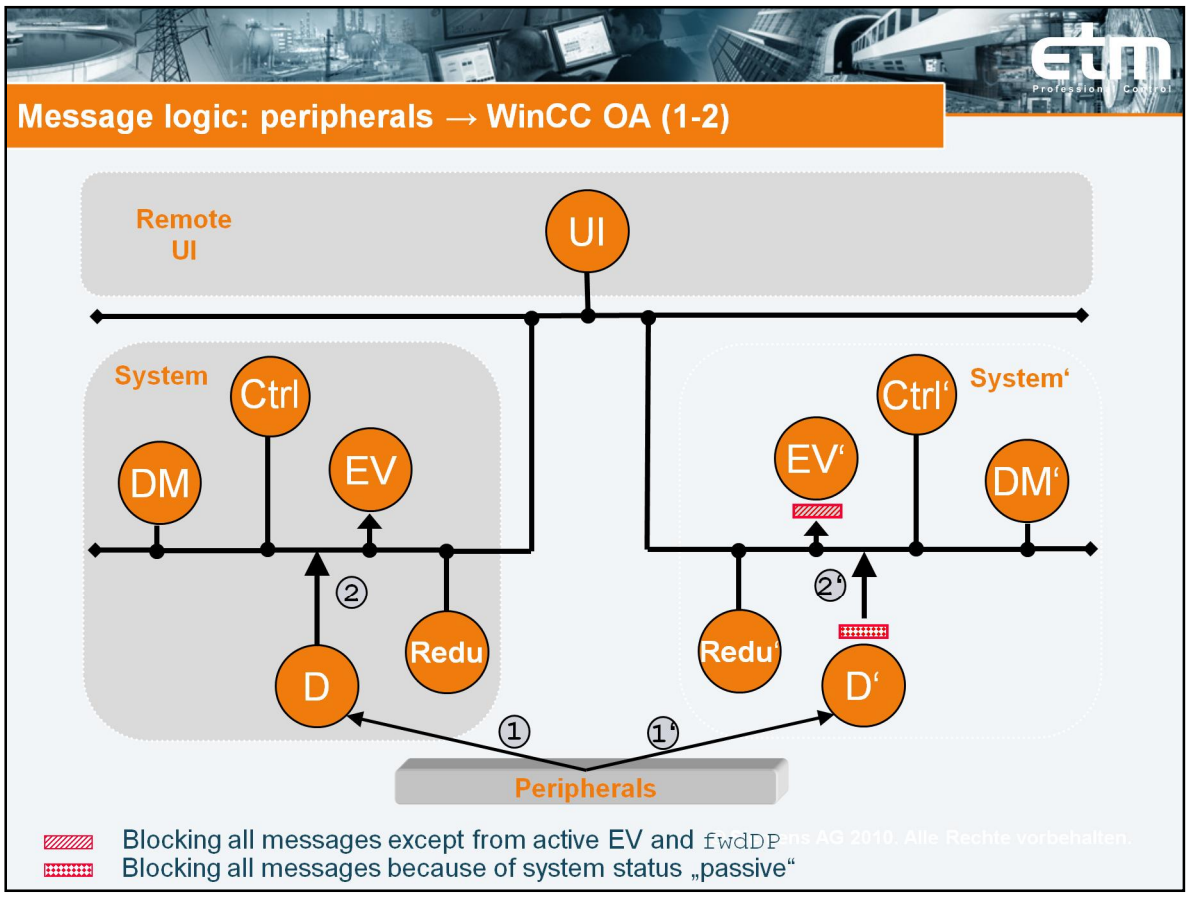
The redundancy is working independent and is not relying on any user interactions or reactions. Nevertheless some commands from the user may be accepted under certain circumstances:

- Definition of a “preferred active” state for one host (this setting is the one with the lowest priority and only taken into consideration if there is the same error state on both systems).

- Immediate manually triggered switching in case of emergency (“forced active” – also carried out when the calculated error state would result in another redu state – that means that the automatic redu-switch because of the error state calculation is disabled → also called manual mode)
- When there was a forced active setting, the user may decide to go back to normal operation (leaving manual mode → going back to automatic mode

Note:

When a redu switch was carried out in a WinCC OA system there will, shortly afterwards, be issued a general query for all drivers (just to play save and retrieve the last recent process states from all existing variables).

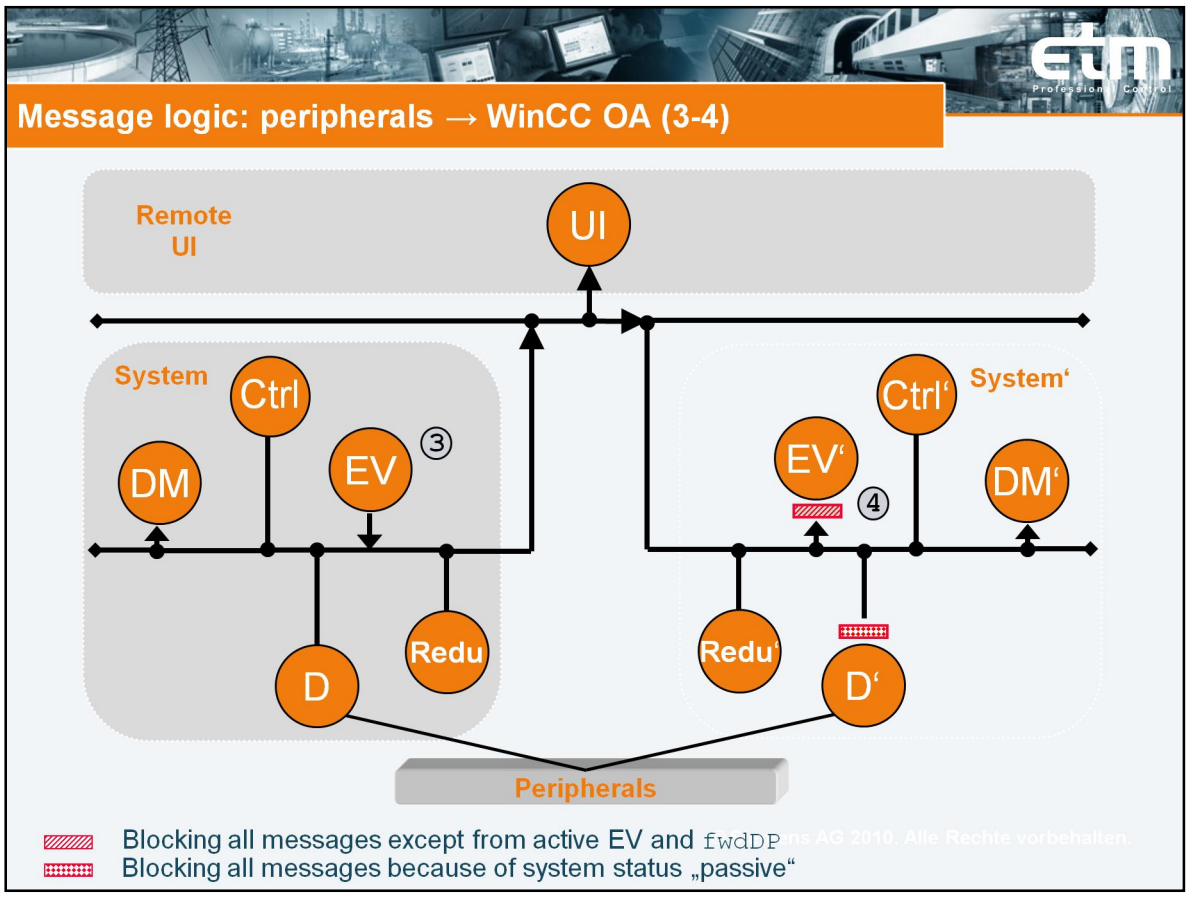


Note:

The above diagram shows the message logic for telegrams from the periphery to WinCC OA.

1) The Process data is changing and transmitted to both drivers (the PLCs do not know whether a WinCC OA server is active or passive) .

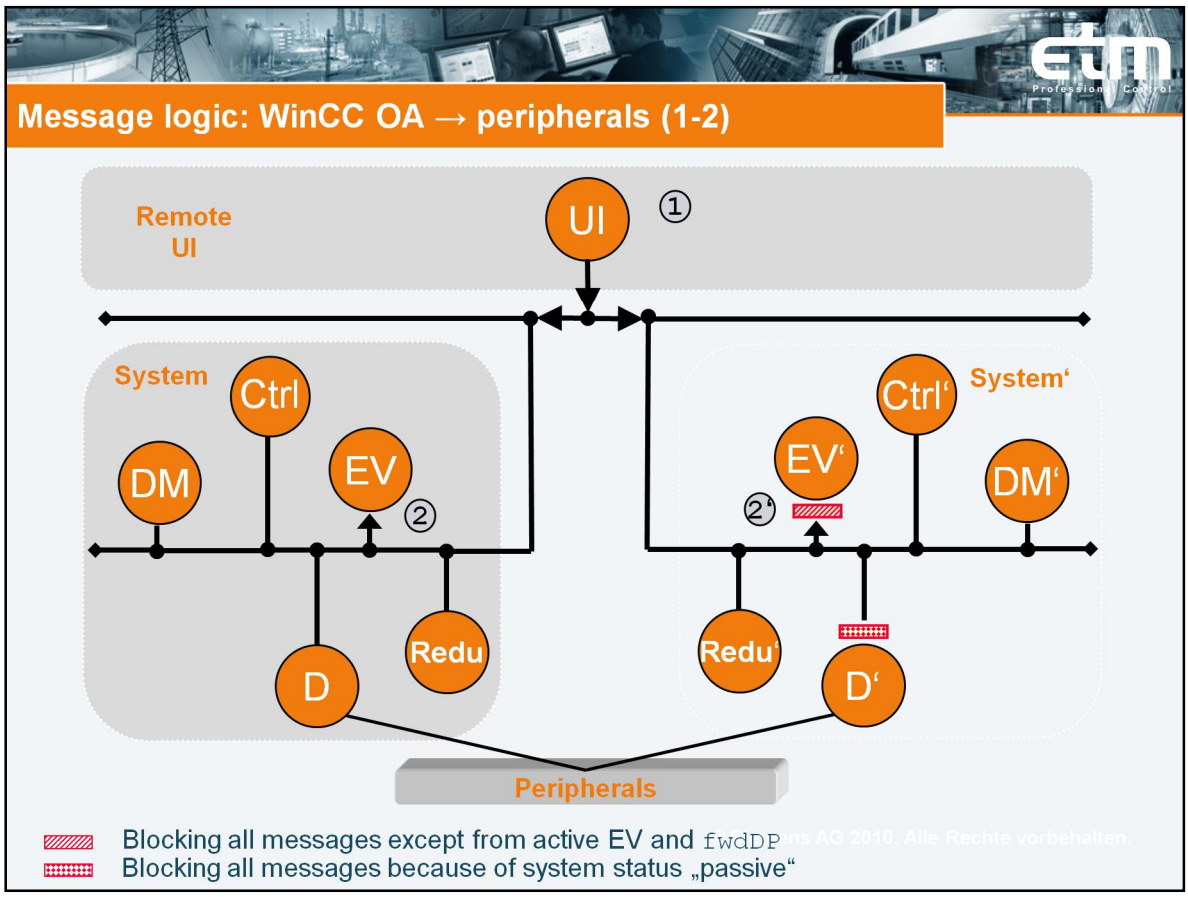
2) **Both** driver processes the new value and will then forward it to its EV-manager; the **active** EV will accept the message, the **passive** one will block the message from its driver.



Note:

3) The **active** EV received the new data point value and will now forward it to its DM, to all manager, that are subscribed to this value (via dpConnect) and to the passive EV.

4) The passive EV will then also forward the message to its (passive) DM.

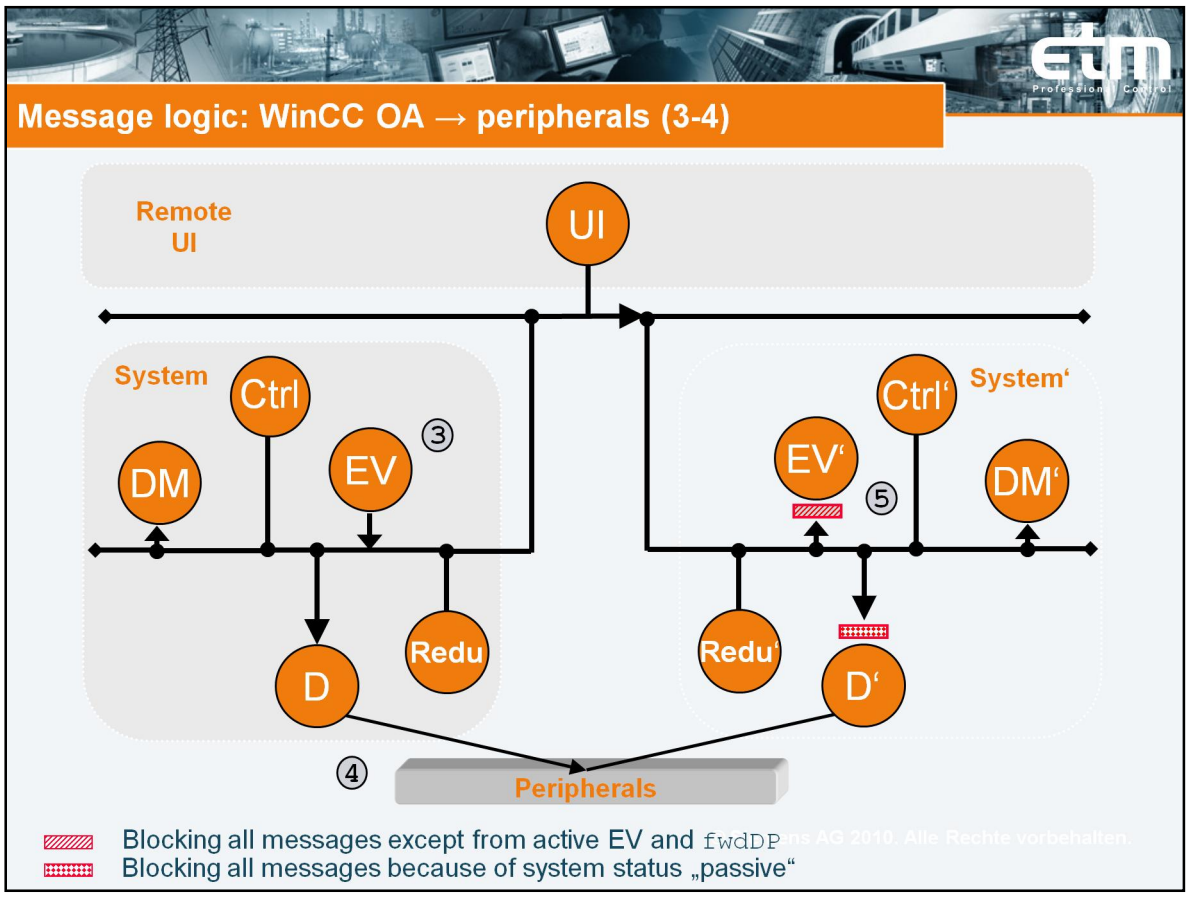


Note:

The above diagram shows the message logic for telegrams from a WinCC OA UI to the periphery. The passive EV will not accept messages from any other manager than his active partner.

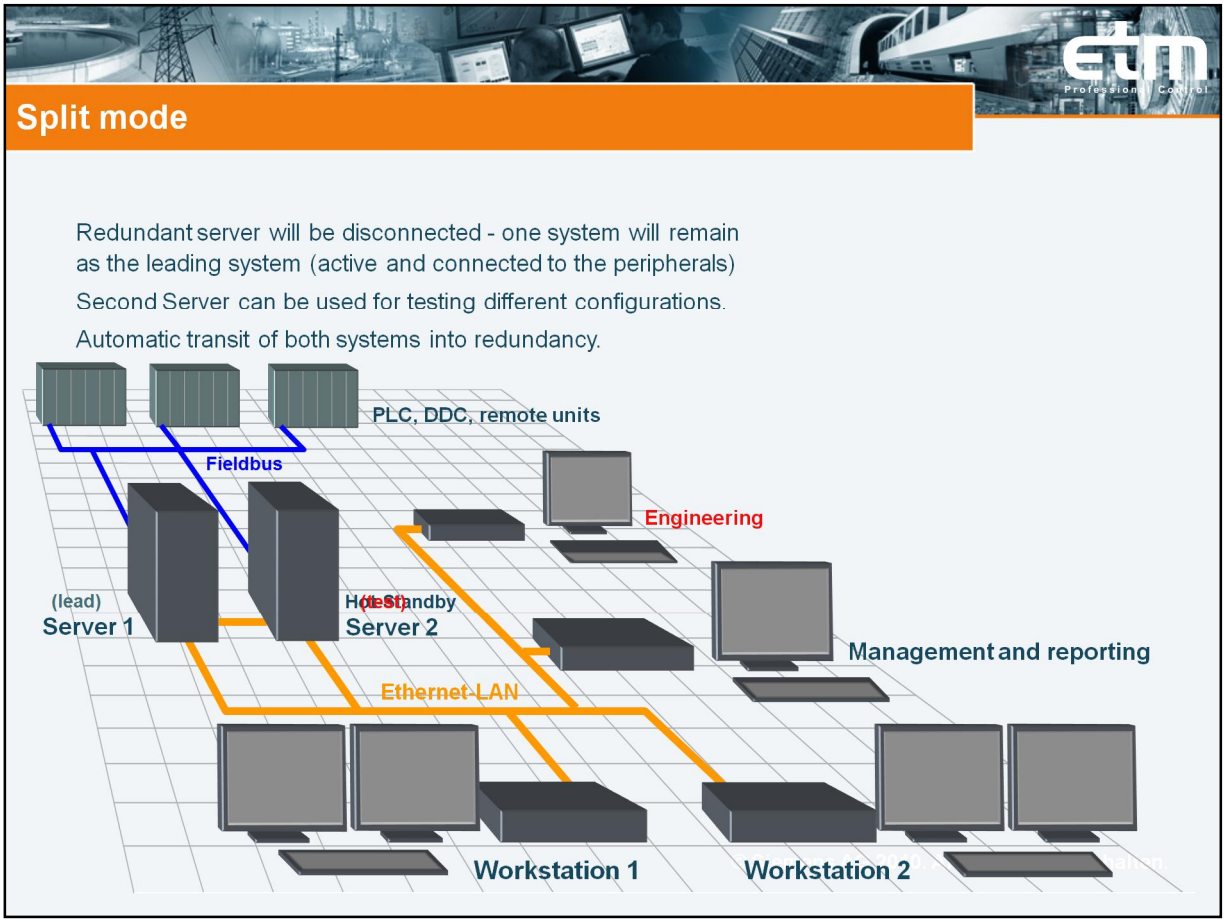
1) The UI will send the command message out to **both** EVs.

2) The **passive** EV is blocking all the messages coming from any UI.



Note:

3) The **active** EV has updated its process states and will send this information to all other manager, that are subscribed to that value. At least it will be sent to its DM, the active driver and the passive EV as well.
4) The **active** driver immediately **sends** this new values to the peripherals.
5) The passive EV will then also forward the message to its (passive) DM.



Note:

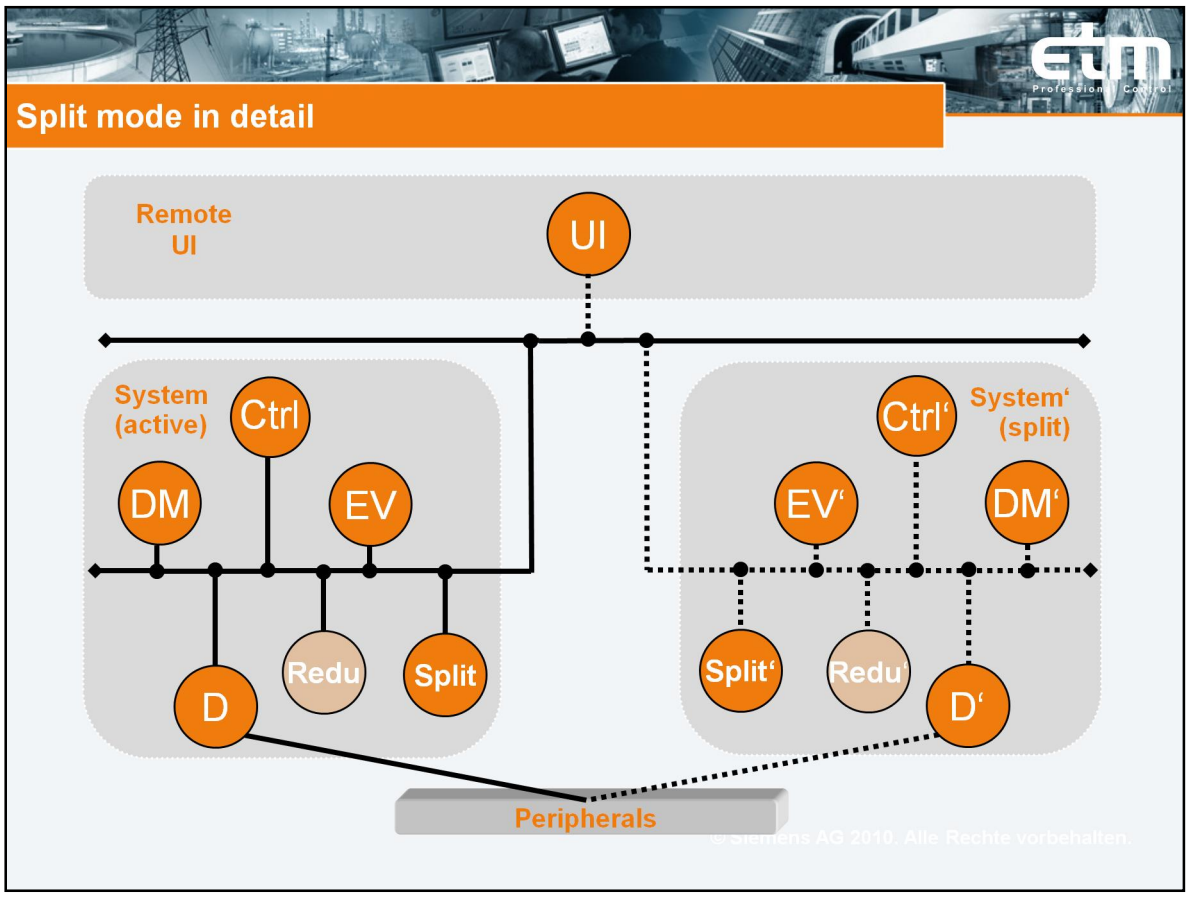
When a new configuration of the project should be tested, this can be done by temporarily disconnect the two redundant servers and bring them in a so called split mode.

Now the splitted system may be used for testing purpose, although it will not send out any commands to the periphery, the split system will still retrieve all the actual changing of process data.

Of course the active system will keep up the normal operation all the time, any other connected UI workstation may notice nothing about the split mode.

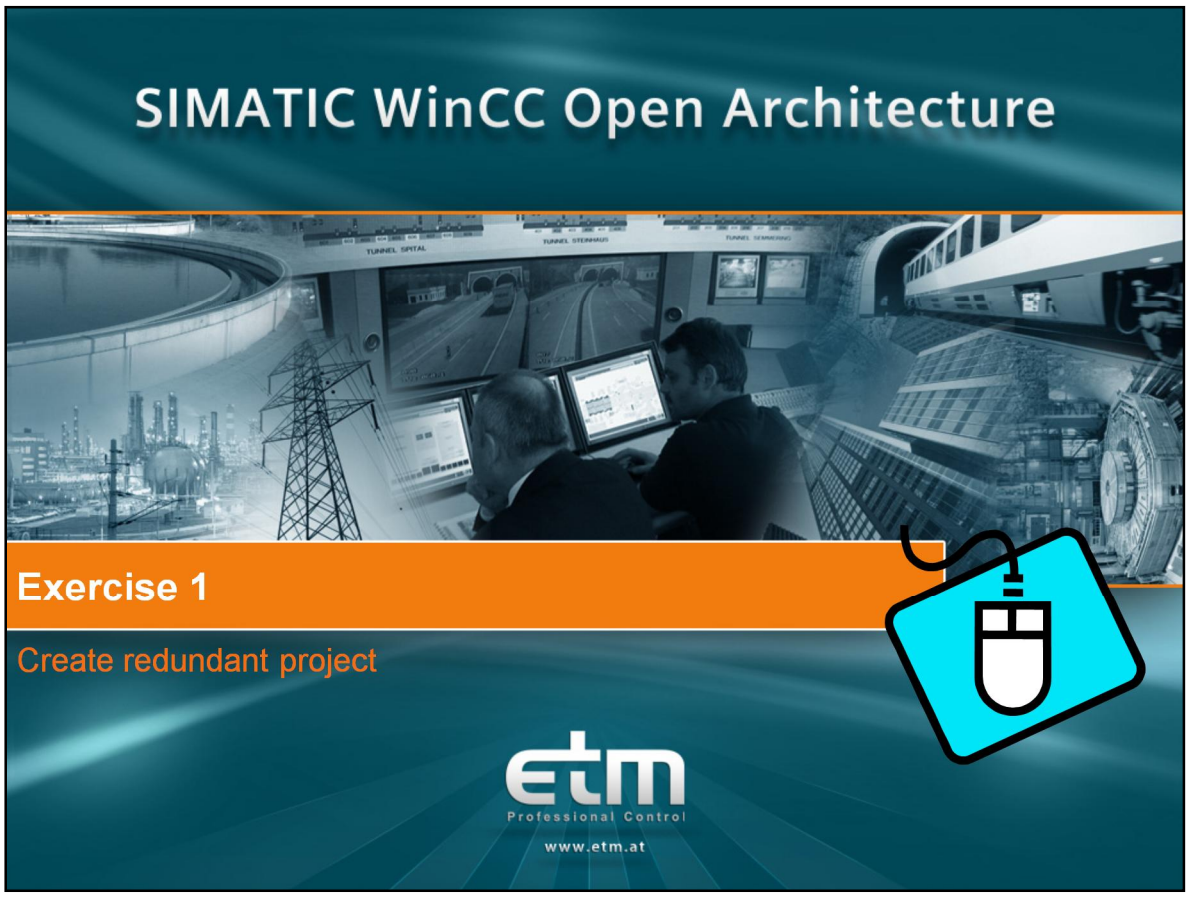
Split mode - features

- Both servers receiving data from process (PLC , Bus….) simultaneously
- Leading system operates normally and supplies workstations with data
- Second server can be disconnected completely and may be used for testing purpose or updating to new releases
- Only some predefined data will be exchanged between the two servers
- Any workstation may be connected to the leading or to the test system
- Automatic return to redundancy mode with selectable server as active system. Retaining original configuration or using new

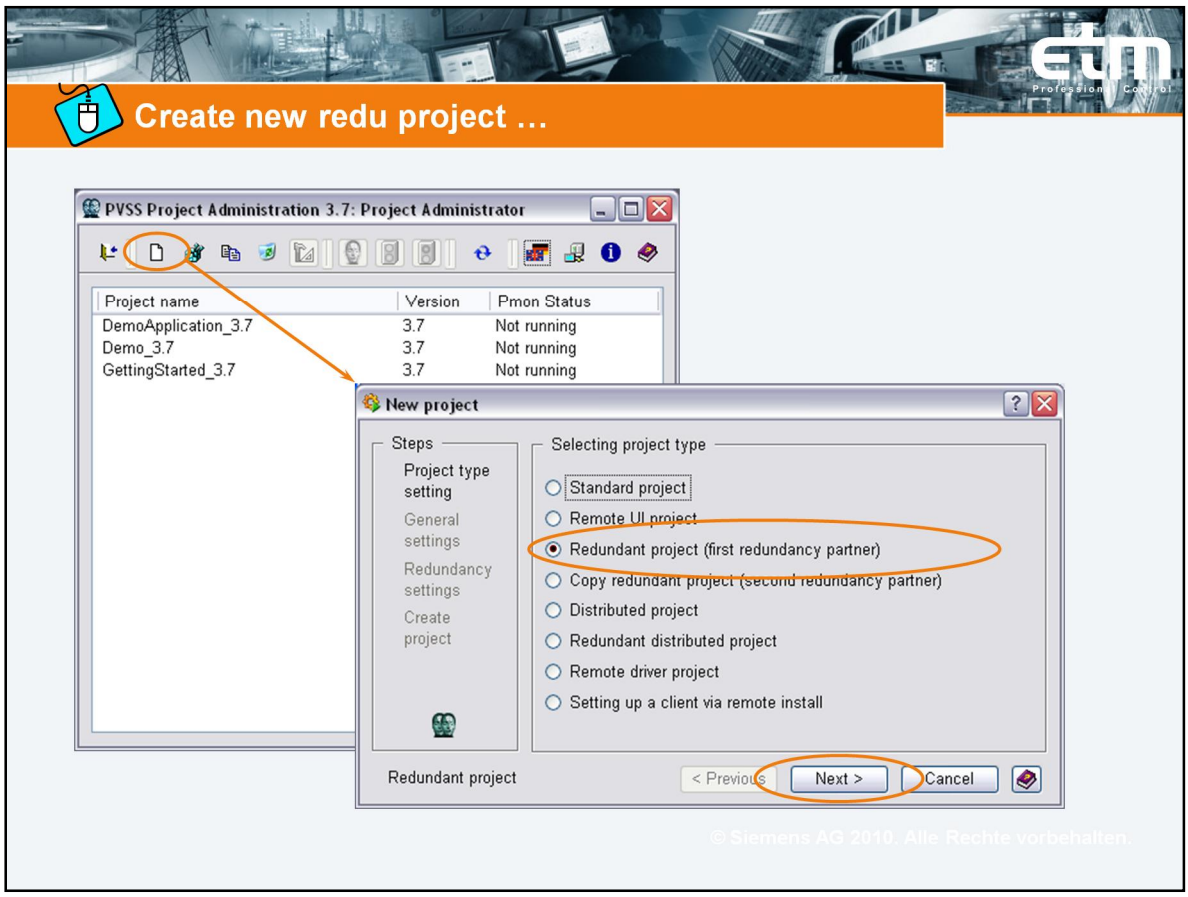


Note:

During split mode both systems are running like they were stand alone systems. The Redu-managers therefore have nothing to do any more and are sent to stand-by mode. The task of the Split-managers will be to keep track of those values, that should be exchanged on demand or continuously between the two systems.


SIMATIC WinCC Open Architecture
Exercise 1
Create redundant project
etm
Professional Control
www.etm.at

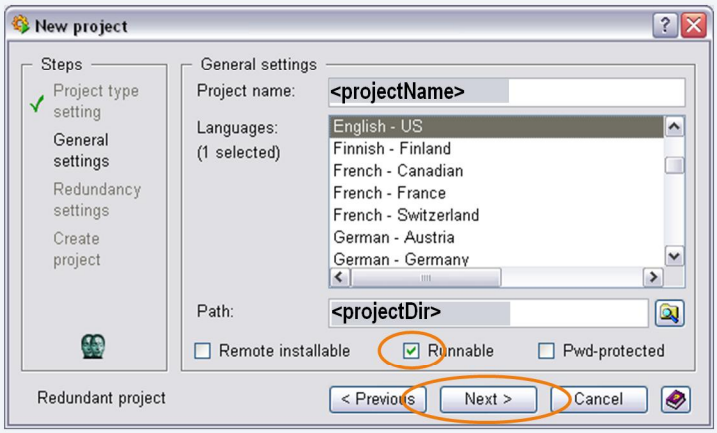

Target of exercise
Tasks:
Configuration of a redundant WinCC OA system
Using single network connection
Using „local" Userinterfaces
© Siemens AG 2010. Alle Rechte vorbehalten.

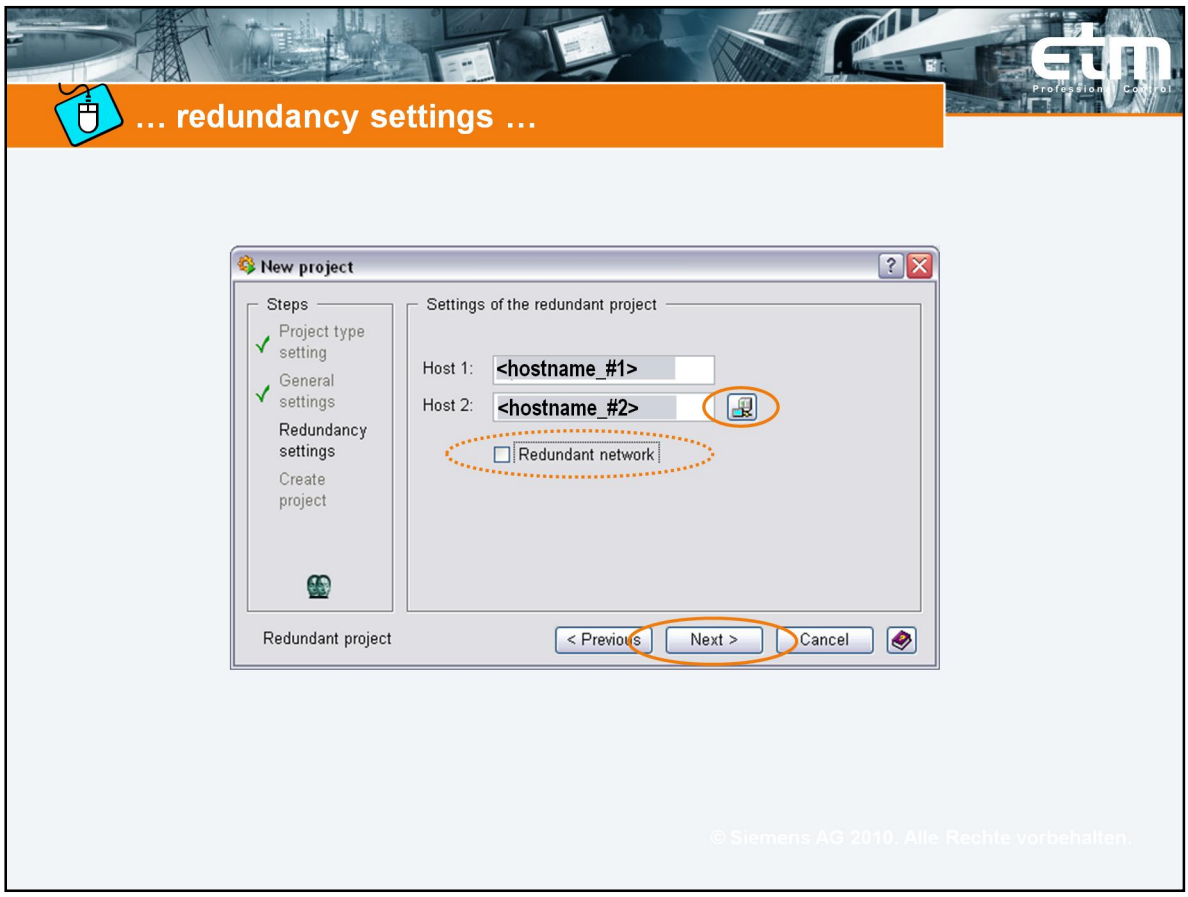


Note:

By using the WinCC OA project administration a new redundant project is created very easily, but nevertheless when an already existing project should be adapted to become a redundant system, the necessary config entries may be added to the config-file manually and also the additional redu manager may be added to the progs-file via the WinCC OA console.

… select name, languages and path …



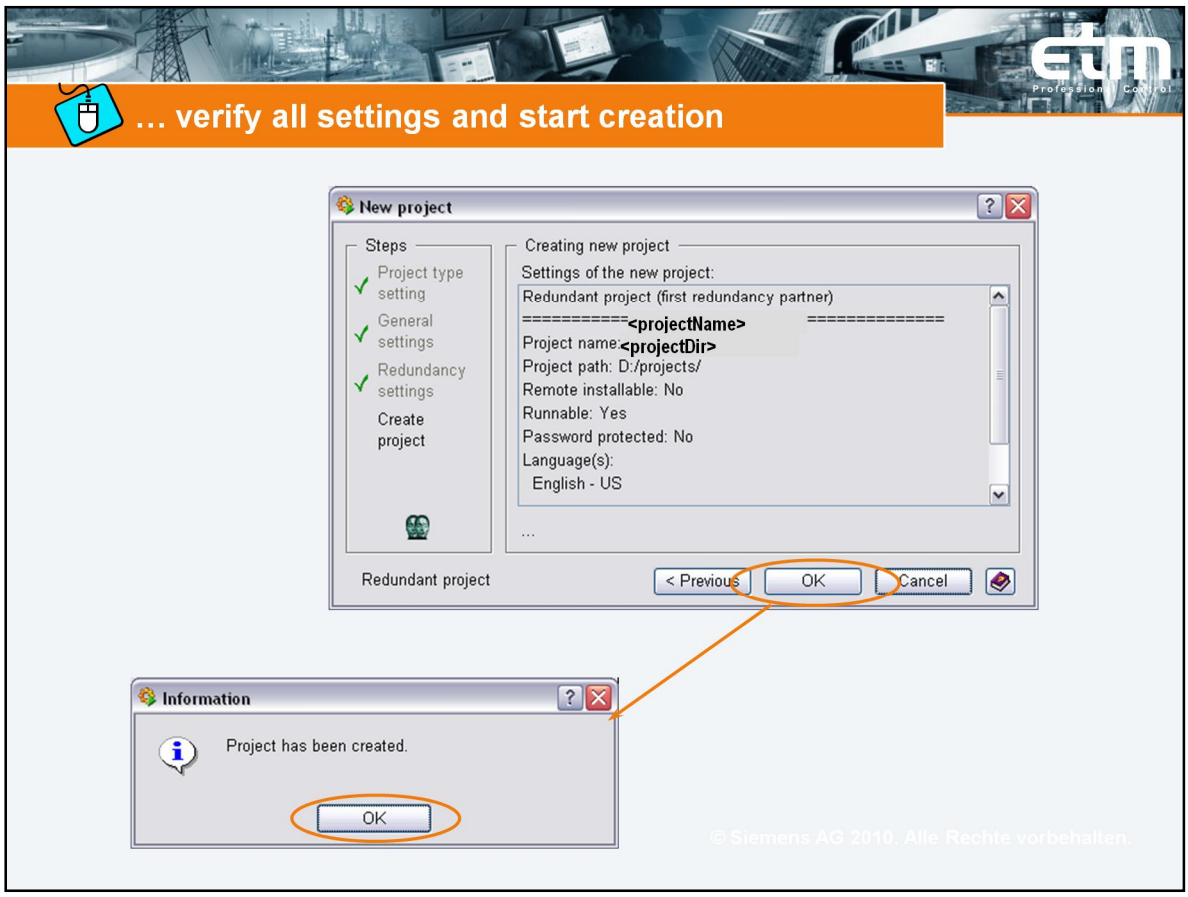


Note:

When having a redundant LAN each server will have two NICs installed. Because each server now will also have two TCP connections for every WinCC OA manager, and in WinCC OA the reverse DNS lookup mechanism is used, we have to make sure that the two different network connections can be identified by a unique name. This is done by introducing a -1 and -2 postfix to the servers hostnames. You have to ensure that your DNS knows about these additional (new) server names or add it manually to the hosts file (on windows machines this file may be found at *"C:/Winnt/system32/drivers/etc/hosts"* - on linux *"/etc/hosts"*).

Attention:

Hostnames that contain a -1 or -2 are not allowed at a redundant WinCC OA system!



Note:

Don't forget that the project folder has to be set up as a shared folder, otherwise the second redu-partner will not be able to copy the project files!

Hint:

All necessary entries to the config file will be done automatically by the setup wizard. If you want to add this manually at an already existing project this may be done in the following way:

```
data = "<host_#1>$<host_#2>"
event = "<host_#1>$<host_#2>"
```

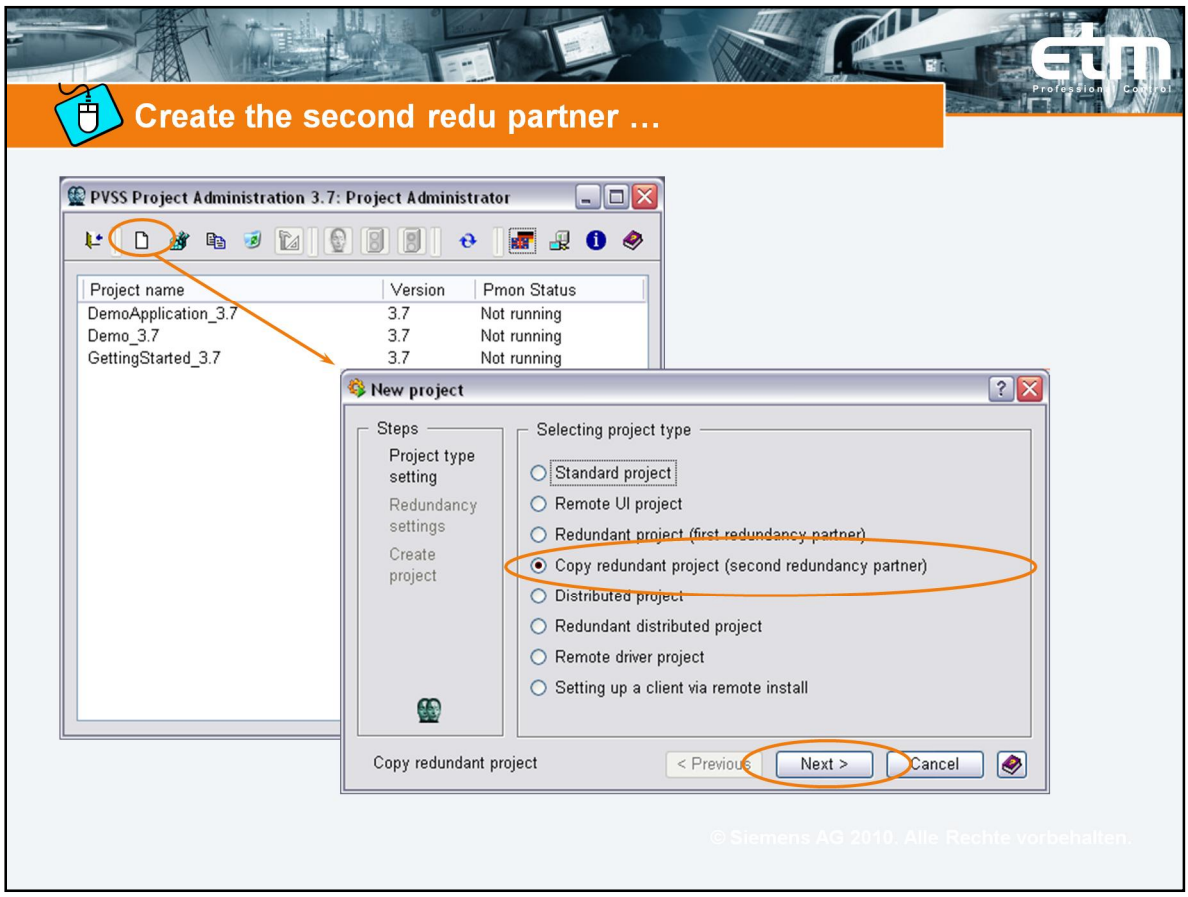
When not using the default port settings the ports have to be added to the hostnames as well.

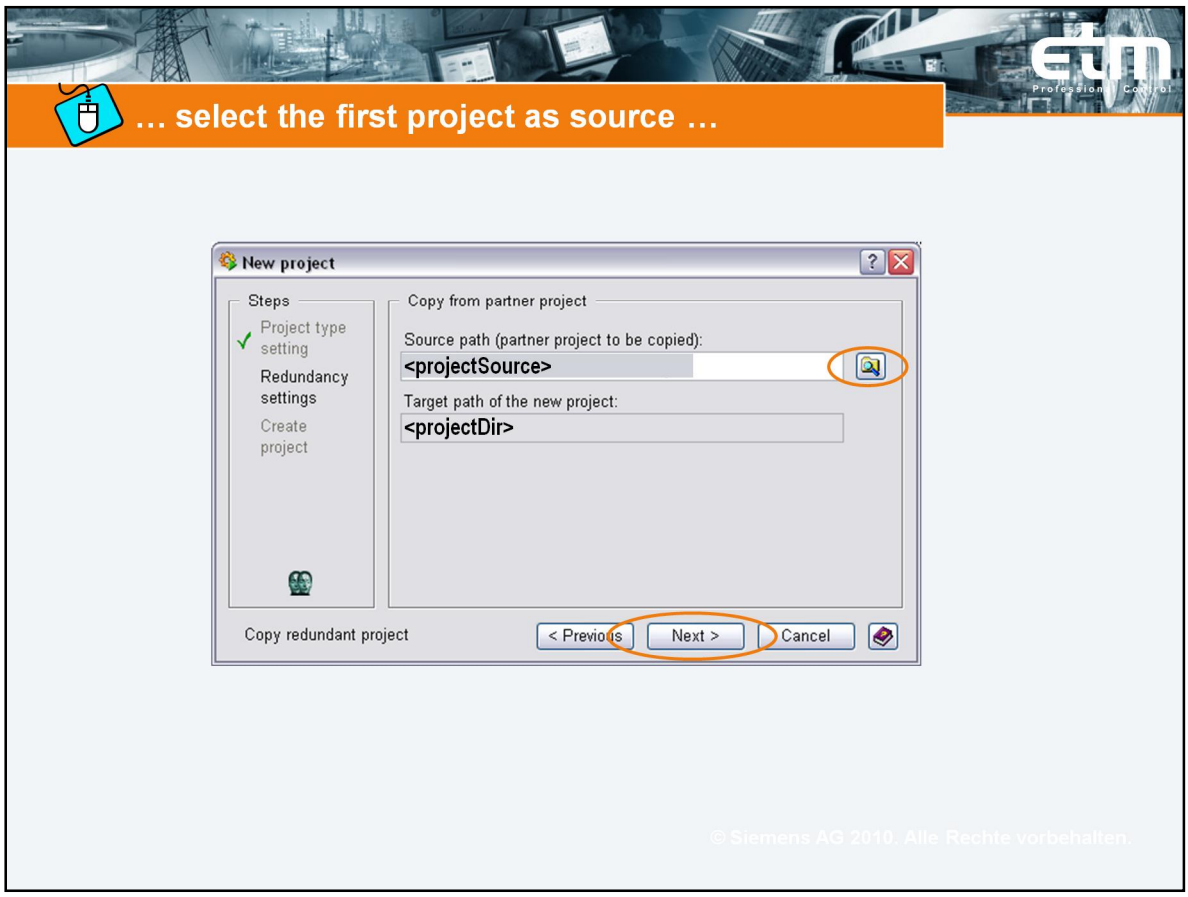
All managers that usually only connects to one of the redundant systems (i.e. Control managers) may also connect to both if this behavior is needed by using the following config entry at the manager-specific part of the config-file:

```
connectToRedundantHosts = 1
```
(Default is 0)

Hint:

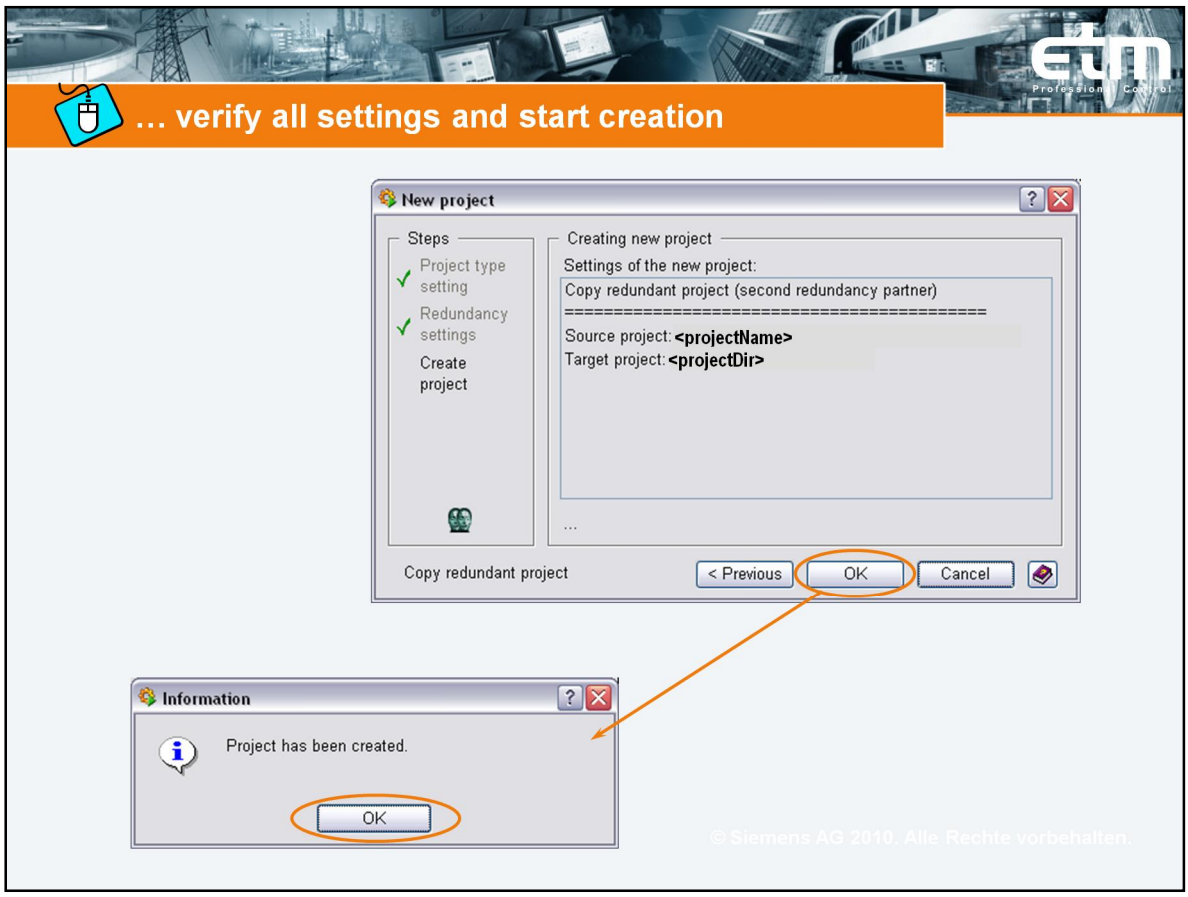
All further settings (timeout at redu-switch, etc.) are applied to a special config file, the `config.redu`. A preconfigured file with useful settings can be found at *<WinCC OAPath>/config*. The adaptation of that file will be covered as an own exercise later on.


Create the second redu partner …
PVSS Project Administration 3.7: Project Administrator
Project name
Version
Pmon Status
DemoApplication_3.7
3.7
Not running
Demo_3.7
3.7
Not running
GettingStarted_3.7
3.7
Not running
New project
Steps
Project type setting
Redundancy settings
Create project
Selecting project type
Standard project
Remote UI project
Redundant project (first redundancy partner)
Copy redundant project (second redundancy partner)
Distributed project
Redundant distributed project
Remote driver project
Setting up a client via remote install
Copy redundant project
< Previous
Next >
Cancel
© Siemens AG 2010. Alle Rechte vorbehalten.



Hint:

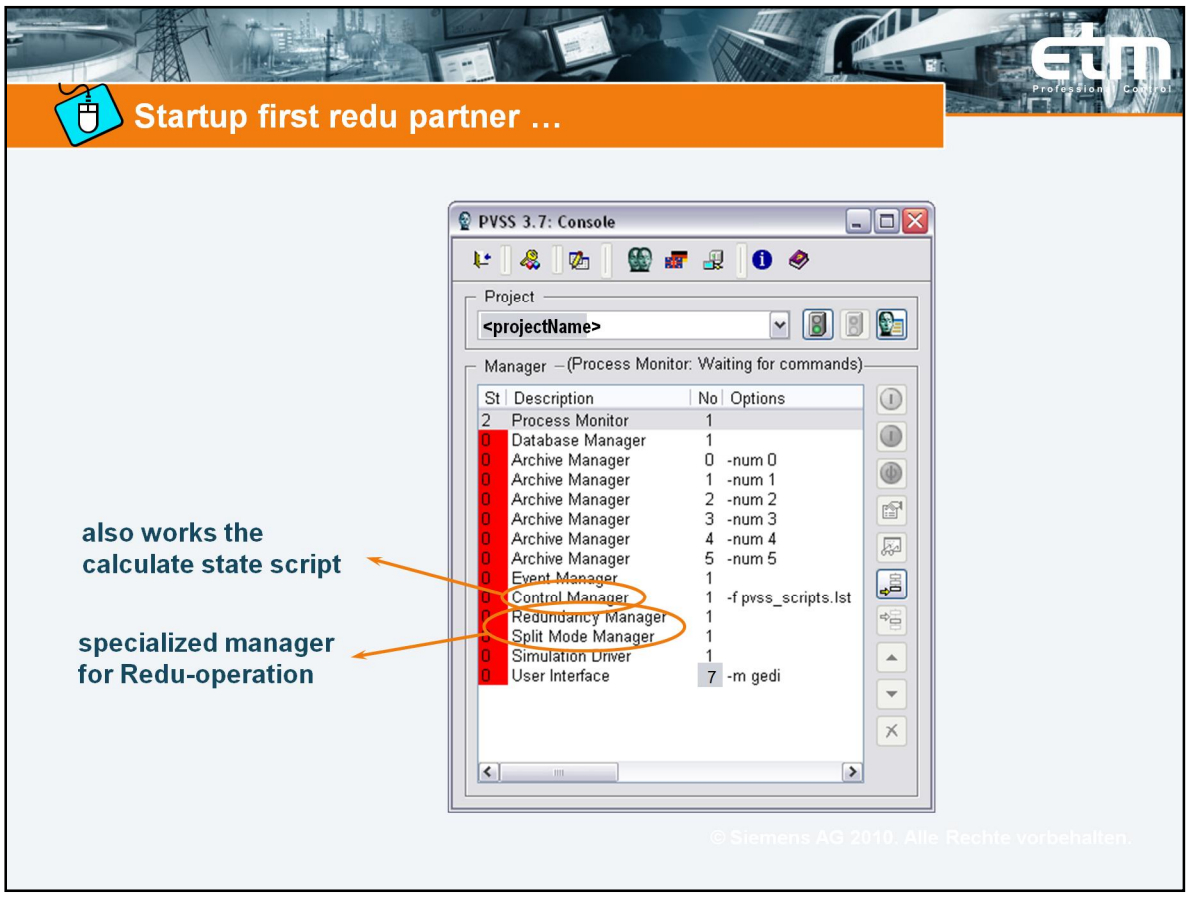
When copying the project to the second redu-partner the target path cannot be chosen freely, but the same path as on the first host will be used. Make sure that the drive-letter used at the first redu-partner is also available on the second system.



Hint:

Make sure, that the source project is not running, because the whole project data will be copied to the other machine (this will include the raima-db)

When the copy is finished don't forget that also this project-folder has to be set up as a shared folder, otherwise the redu-partner will not be able to access the data files for a possible alignment later wards.



Note:

The necessary manager for redu operation (Redu und Split), as well as a script that calculates the error state of a system in the background (`calculateState.ctl`) are already added to the console.

Hint:

The order of managers starting up shall not be changed, at least from the DM-manager until the Ctrl-manager. Project specific managers should be added to the console after the Ctrl-manager. A “Local" UI (directly started on a server) will get a manager number starting at 7 (this is only a default and may be changed in the config file). **Remote UIs should in any case use fixed manager numbers (the –num option) ranging from 1 to 6.**

Hint:

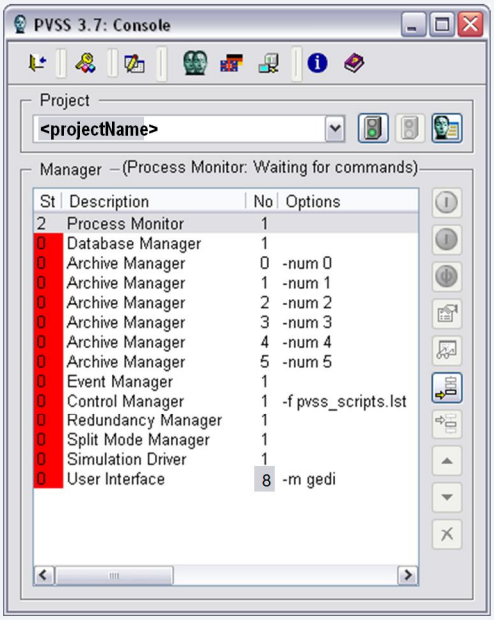
Before starting up the two projects, please make sure that the two servers have synchronous time settings.

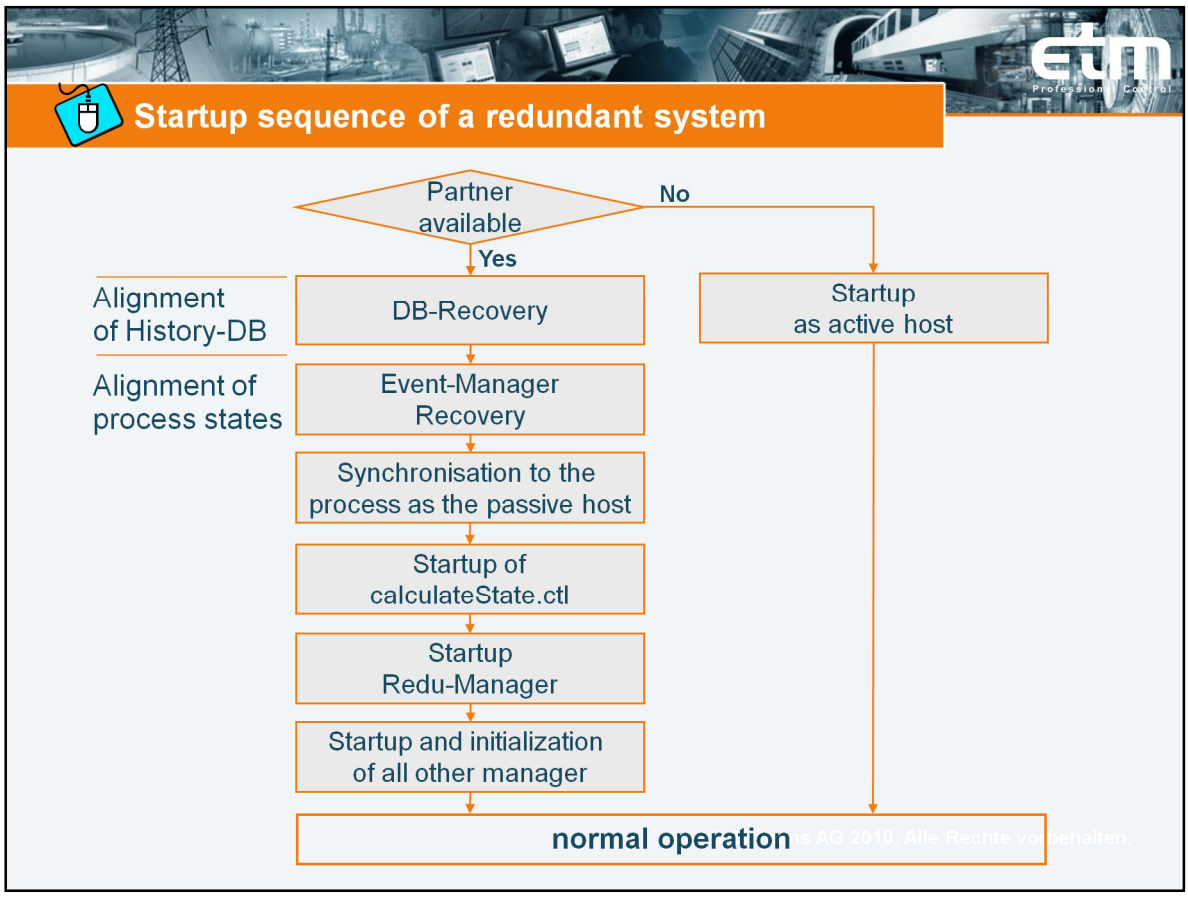
The usage of a time synchronization software is highly recommended.

… and the second redu partner

same project name ...

... and same manager as at the first redundancy partner





Note:

When starting up a redundant project the user have to start both systems, one after the other. The second host that comes up will then be automatically aligned to the actual process data of the first one started.

When both systems are up and running in sync the calculated error state will decide which system then will become the „active“ or the „passive“ one. When both servers are running with the same error state (this should be the most usual case), than the first one started will keep active and the second one will join in as the passive (hot-standby) machine.

Note:

Hints to the log-file entries on the FIRST redundant system:

In the following a short overview is given about the log entries that are generated at startup of the **first** redu partner:

<time> ... recent timestamp
<host_#1> ... first redundancy partner
<host_#2> ... second redundancy partner

1) The DM starts up and tries to find his redundant partner:

```
WCCILdata   (0),<time>, REDU, INFO, 3, Trying to connect to (SYS: 0 Data -num 0 CONN: 2)
@ <host_#2> (may take some time if host is unreachable)
WCCILdata   (0),<time>, REDU, INFO, 0,  , Could not connect to other replica, starting
active.
```

2) The archive managers are starting up and getting initialized by the local DM

3) The EV is starting and connecting to the local DM, gets initialized and starts operation as the active system:

```
WCCILevent  (0),<time>, SYS,  INFO, 3,  Trying to connect to, (SYS: 0 Data -
num 0 CONN: 1) @ <host_#1>:4897
WCCILevent  (0),<time>, REDU, INFO, 0,  , Status change to active
```

4) The Redu- and Split-manager, as well as the drivers and the Ctrl-managers are starting up and being initialized by the local DM. This is the point where the first system is fully operational. In our example there is a local UI started in addition

5) The first (local) UI starts up and first of all it tries to connect to both redundant partner. Of course it will, at the moment, only find the local system, the other one is not started yet. After that all the manager of the first redu partner are up and running.

```
WCCOAui     (0),<time>, SYS,  INFO, 3,  Trying to connect to, (SYS: 0 Data -
num 0 CONN: 1) @ <host_#1>:4897$<host_#2>:4897
WCCOAui     (0),<time>, SYS,  INFO, 4,  Connected to, (SYS: 0 Data -num 0
CONN: 1) @ localhost
WCCOAui     (7),<time>, REDU, INFO, 0,  , Closing connection to redundant data
during initialization
WCCOAui     (7),<time>, SYS,  INFO, 3,  Trying to connect to, (SYS: 1 Event -
num 0 CONN: 1) @ <host_#1>:4998
WCCOAui     (7),<time>, REDU, INFO, 0,  , Closing connection to redundant
event during initialization
```

Note:

Hints to the log-file entries on the SECOND redundant system :

In the following a short overview is given about the log entries that are generated at startup of the **second** redu partner:

<time> ... recent timestamp
<host_#1> ... first redundancy partner
<host_#2> ... second redundancy partner

1) The DM starts up and tries to find his redundant partner. When the redu partner can be contacted, it is clear that there is already an active system and an alignment to the current process data has to be initiated. The alignment is done for all historical data changes that had been archived by the first redu partner so far (this may take some time in the case the second partner was offline for a longer period of time).

```
WCCILdata   (0),<time>, REDU, INFO, 3, Trying to connect to (SYS: 0 Data -num 0
CONN: 2) @ <host_#1> (may take some time if host is unreachable)
WCCILdata   (0),<time>, SYS,  INFO, 4,  Connected to, (SYS: 0 Data -num 0 CONN: 1)
@ <host_#1>
WCCILdata   (0),<time>, REDU, INFO, 0,  , Sending recovery request to other replica
WCCILdata   (0),<time>, REDU, INFO, 0,  , Recovery request accepted, sending file
list request
WCCILdata   (0),<time>, REDU, INFO, 0,  , File transfer request sent
WCCILdata   (0),<time>, REDU, INFO, 0,  , Recovery completed, starting passive
```

2) The archive managers are starting up and getting initialized by the local DM

3) The EV is starting, connecting to the local DM and gets initialized. Because the local DM is running in passive mode the EV will also be initialized in passive mode and request the recent process view from his active partner

```
WCCILevent          (0),<time>, SYS,  INFO, 3, Trying to connect to, (SYS: 0 Data -
num 0 CONN: 2) @ <host_#2>:4897
WCCILevent          (0),<time>, REDU, INFO, 0, , Starting buffering for passive
recovery
```

4) The Redu- and Split-manager, as well as the drivers and the Ctrl-managers are starting up and being initialized by the local DM. This is the point where the second system is fully operational as well. In our example there is a local UI started in addition.

5) The second (local) UI starts up and first of all it tries to connect to both redundant partner. Unlike at the first UI there are both systems now up and running. The UI will start to connect to both systems but only the active connection will remain open, the connection to the passive system is closed shortly afterwards.

```
WCCOAui     (0),<time>, SYS,  INFO, 3,  Trying to connect to, (SYS: 0 Data -num 0 CONN: 1) @ <host_#1>:4897$<host_#2>:4897

WCCOAui     (0),<time>, SYS,  INFO, 4,  Connected to, (SYS: 0 Data -num 0 CONN: 1) @ <host_#1>
WCCOAui     (0),<time>, SYS,  INFO, 4,  Connected to, (SYS: 0 Data -num 0 CONN: 2) @ <host_#2>
WCCOAui     (8),<time>, REDU, INFO, 0,  , Closing connection to redundant data during initialization
WCCOAui     (8),<time>, SYS,  INFO, 3,  Trying to connect to, (SYS: 1 Event -num 0 CONN: 1) @ <host_#1>:4998
WCCOAui     (8),<time>, SYS,  INFO, 4,  Connected to, (SYS: 1 Event -num 0 CONN: 1) @ <host_#1>
WCCOAui     (8),<time>, REDU, INFO, 0,  , Closing connection to redundant event during initialization
```

Note:

Hints to the log-file entries on the FIRST redundant system during the data alignment:

In the following a short overview is given about the log entries of the **first** system, when the second system is initiating the data recovery.

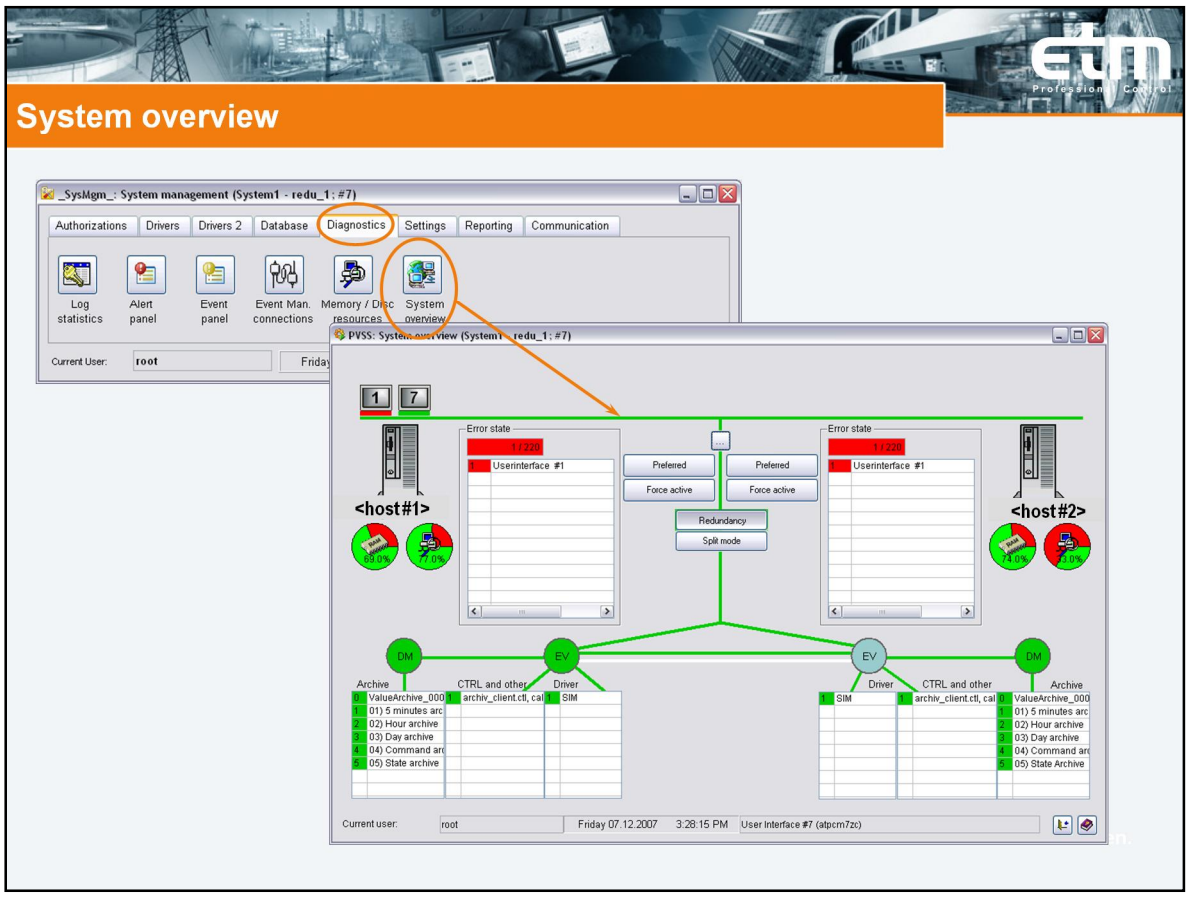
<time> ... recent timestamp
<host_#1> ... first redundancy partner
<host_#2> ... second redundancy partner

1) The DM on the first redu partner gets an incoming recovery request of the second redu partner and will inform his local EV about that. To get a consistent process view on both systems, the EV-manager will have to buffer all incoming messages as long as the DM recovery of the historical data is lasting.

```
WCCILdata          (0),<time>, SYS,  INFO, 4, Connected to, (SYS: 0 Data -num 0
CONN: 2) @ <host_#2>
WCCILdata          (0),<time>, REDU, INFO, 0, , Got recovery request, forwarding
to event.
WCCILevent         (0),<time>, REDU, INFO, 0, , Accepting recovery request,
sending ACK to data
WCCILevent         (0),<time>, REDU, INFO, 0, , Starting buffering for active
recovery
WCCILdata          (0),<time>, SYS,  INFO, 0, , DataManager, stopBgMgr, Stopping
WCCILdatabg
WCCILdata          (0),<time>, REDU, INFO, 0, , Copying files finished
successfully,      leaving mirror mode
WCCILdata          (0),<time>, IMPL, INFO, 0, , DataManager, startBgMgr,
(Re)starting WCCILdatabg
WCCILdatabg        (1),<time>, SYS,  INFO, 6, Initialization by Data Manager
finished
```

2) After the recovery of the database is finished the both systems are in sync to each other and the active EV manager can build up a connection to its passive partner and stop buffering. From now on the process data on the passive system will be updated in realtime.

```
WCCILevent         (0),<time>, REDU, INFO, 0, , Link 0 up, enable connect to
redundant peer
WCCILevent         (0),<time>, SYS,  INFO, 4, Connected to, (SYS: 1 Event -num 0
CONN: 2) @ <host_#2>
WCCILevent         (0),<time>, REDU, INFO, 0, , Recovery finished successfully
```

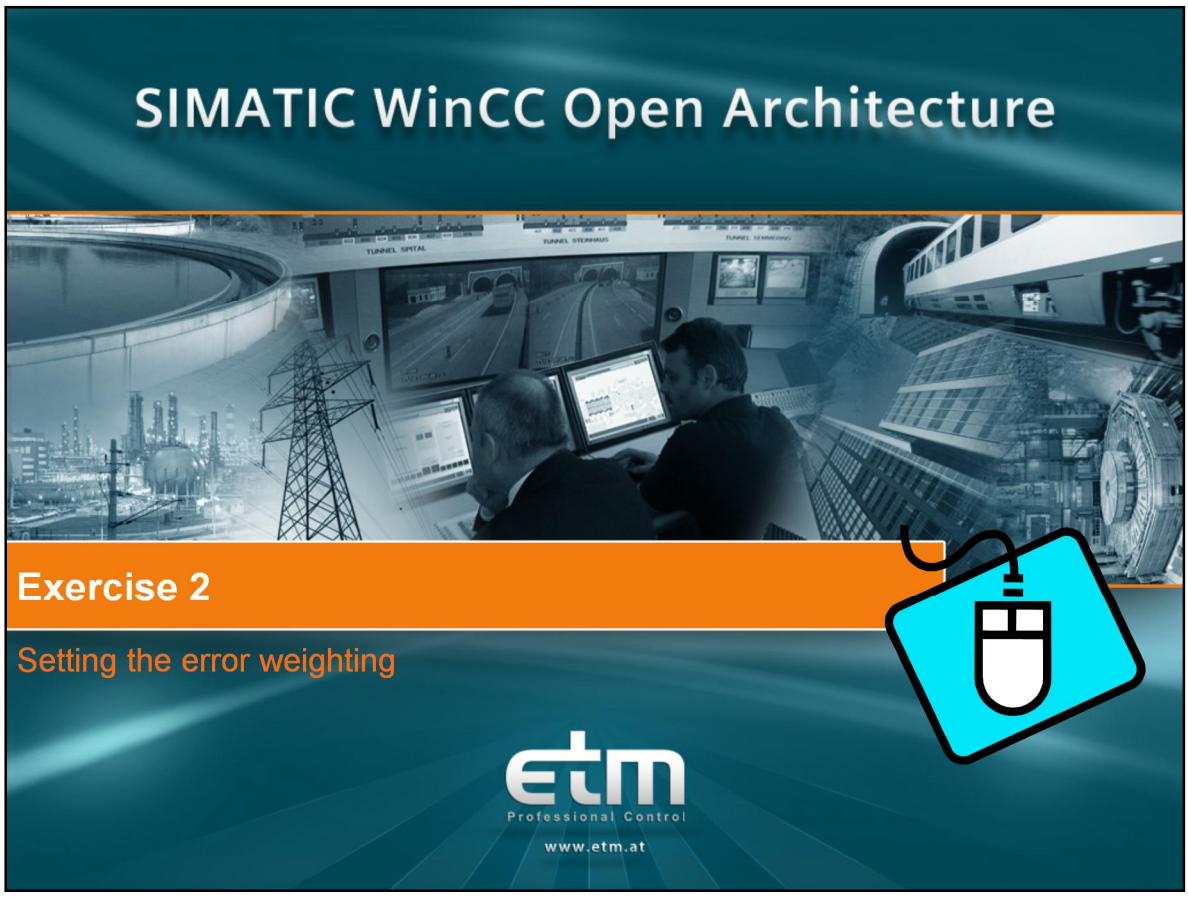


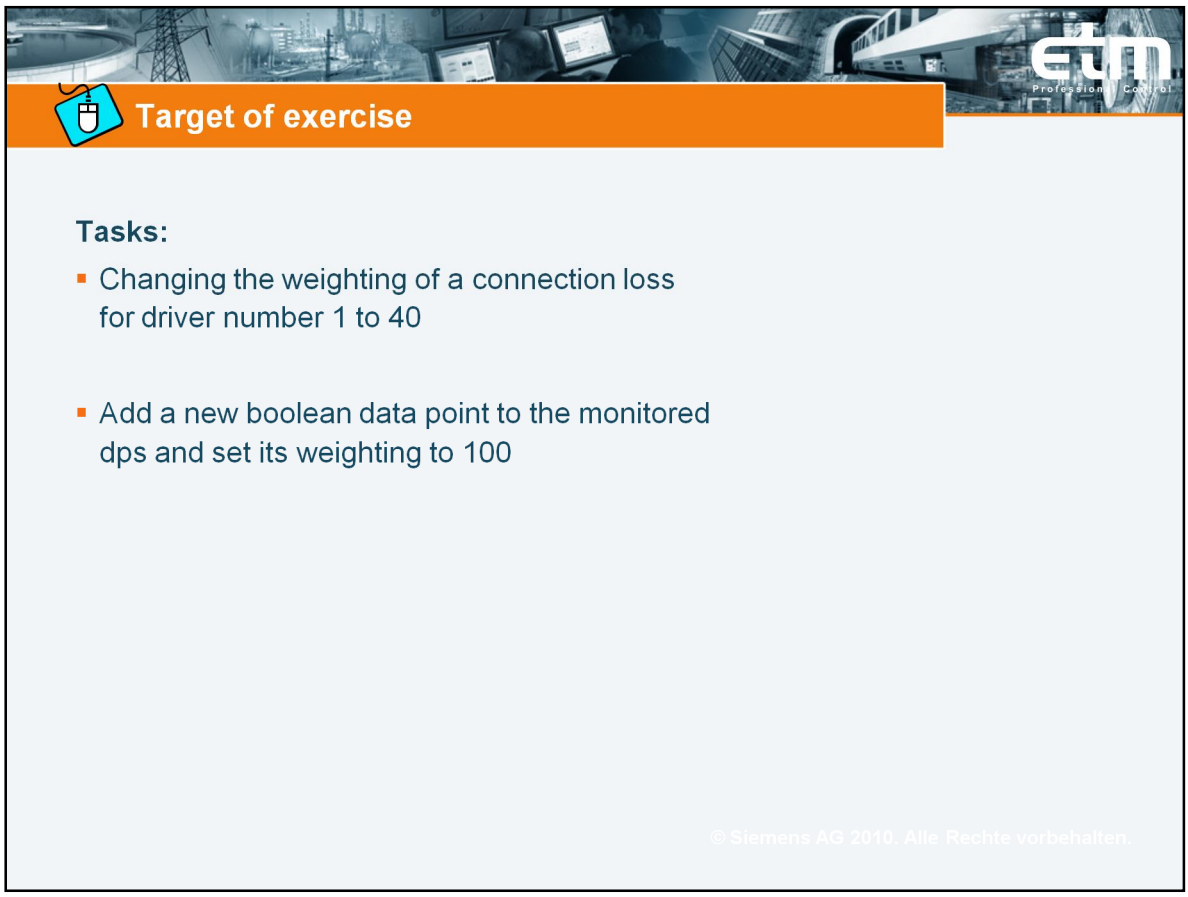
Note:

At the system overview panel the detailed states of both redundant partner (host_#1 and host_#2) can be monitored. At the above screenshot the left system is the active one (to be recognized because of the green EV-manager symbol)

In the upper Area of the panel the currently connected UIs will be displayed. In the tables below the EV symbol the local connected managers are listed (grouped by Ctrl, Drivers, Archives and maybe Dist).

Directly aside the drawings of the two hosts the actual error state of each system can be monitored (the error state is shown in the following way: “current error state” / “maximum error state possible” i.e. the sum of all parameterized weightings). The table directly below lists up all current errors in detail (weight and description of error). The weighting of the monitored errors may be parameterized by right clicking on that table or using the above button with the three dots on it. There are also additional buttons implemented to change the preferred redu system or set a distinct system to forced active mode. Also a button can be found for sending the redundant system into the split mode and another one to come back to redundancy again.

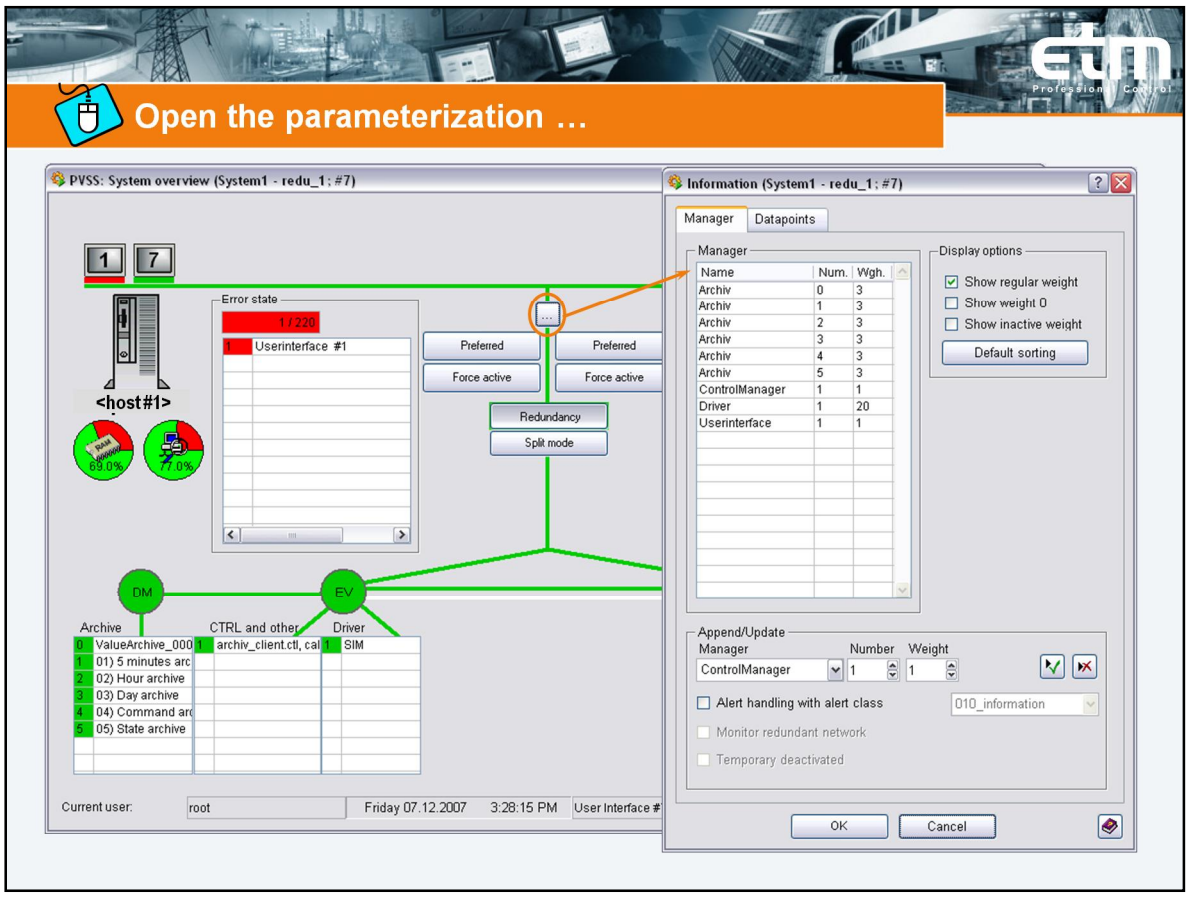

SIMATIC WinCC Open Architecture
Exercise 2
Setting the error weighting
etm
Professional Control
www.etm.at



Target of exercise

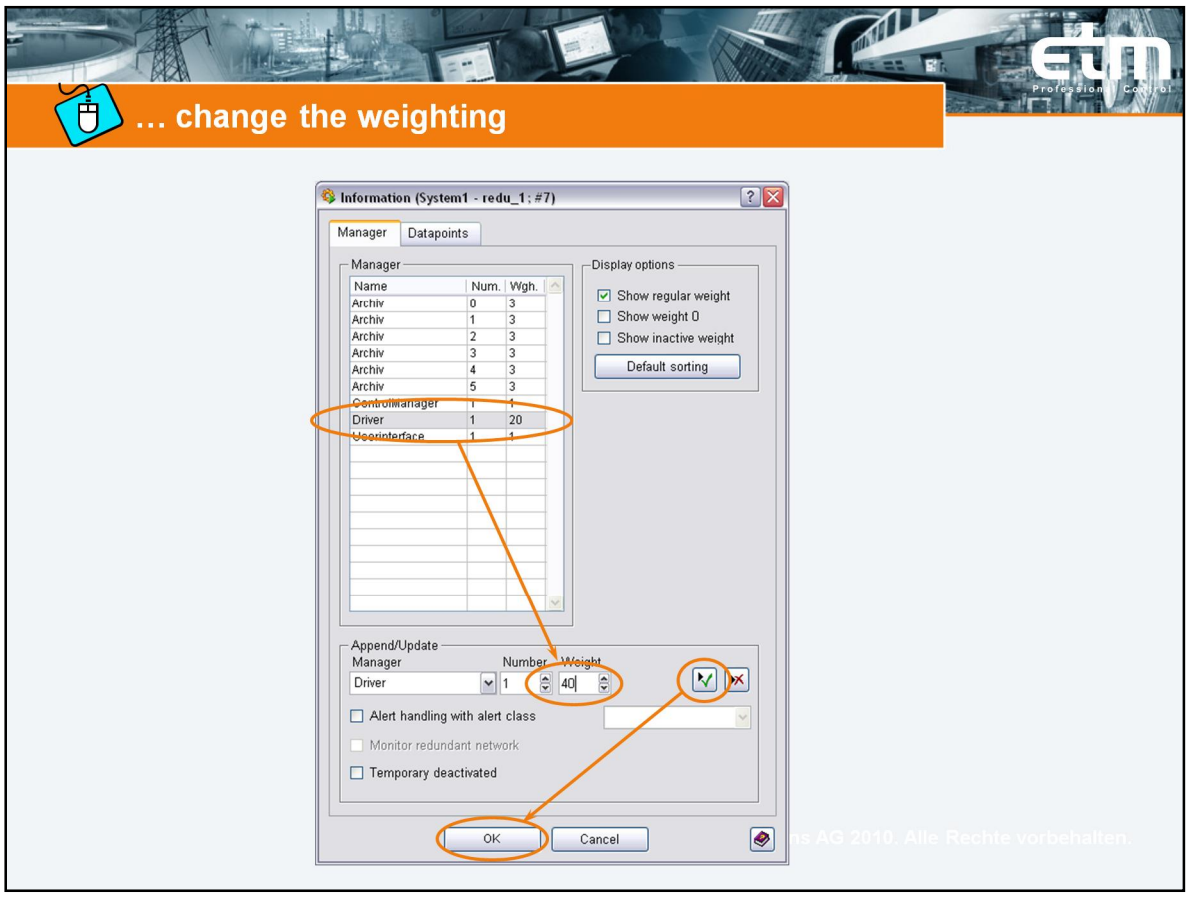
Tasks:

- Changing the weighting of a connection loss for driver number 1 to 40
- Add a new boolean data point to the monitored dps and set its weighting to 100



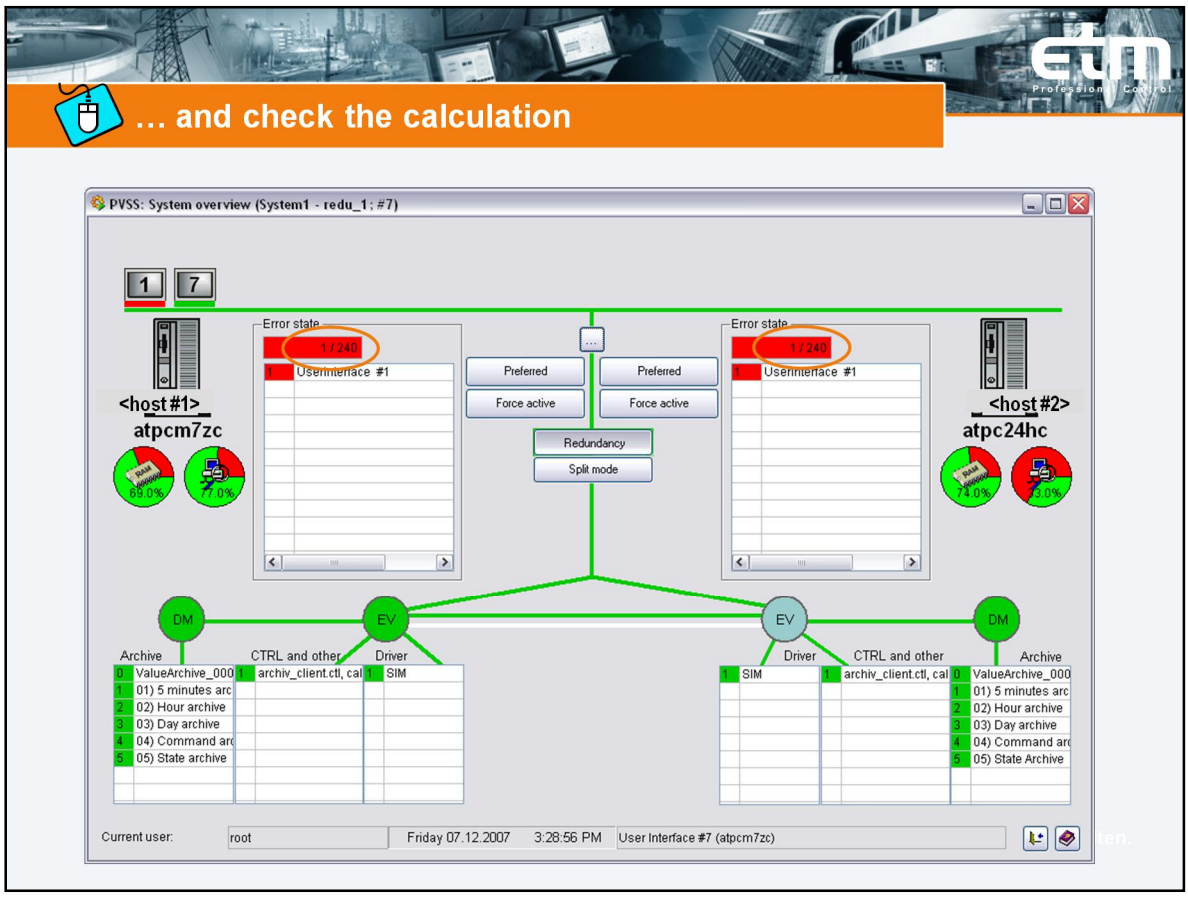
Note:

The parameterization panel for the error weighting setting may also be opened by right clicking on of the shown tables (error state, archive, driver, Ctrl, etc.).



Note:

A loss of connection to the driver with the number one, is now contributing 40 points to the error state of the respective system.

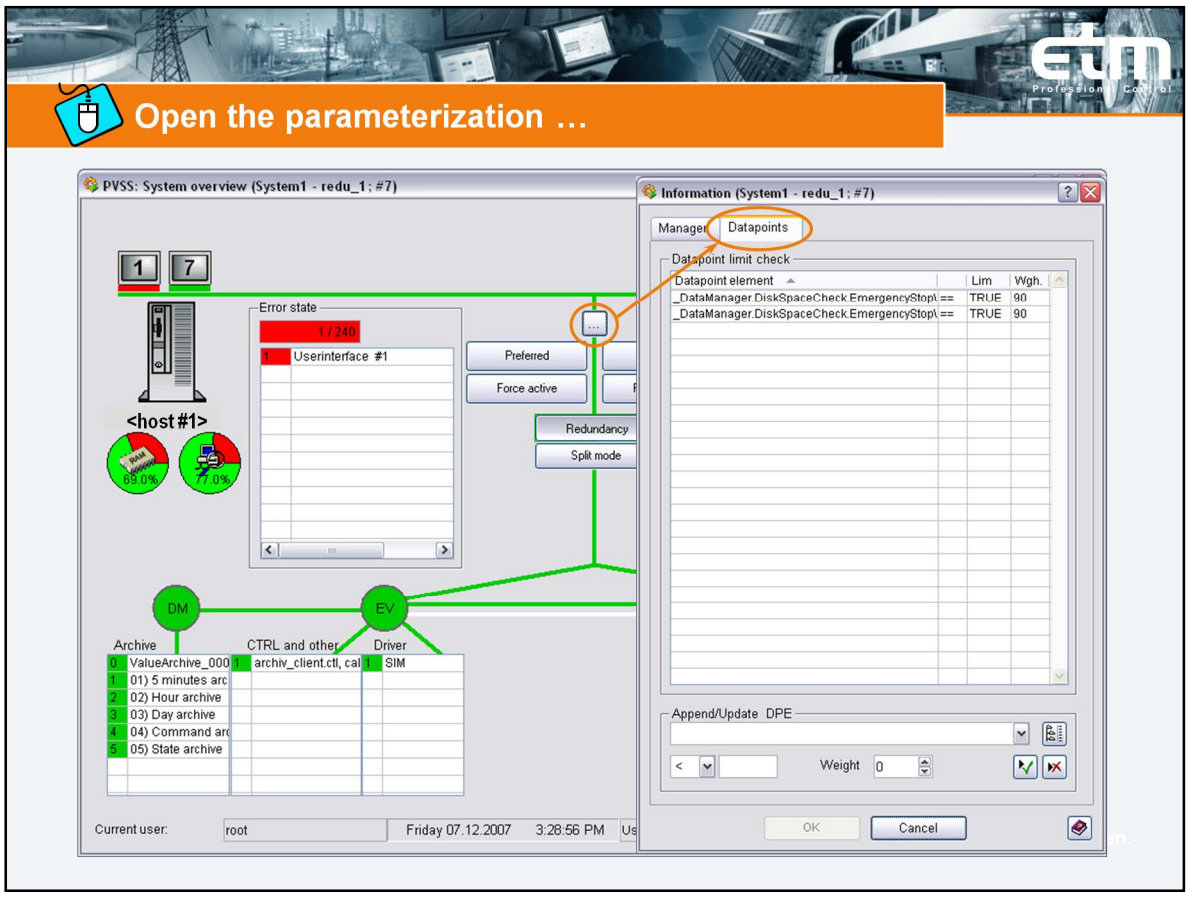


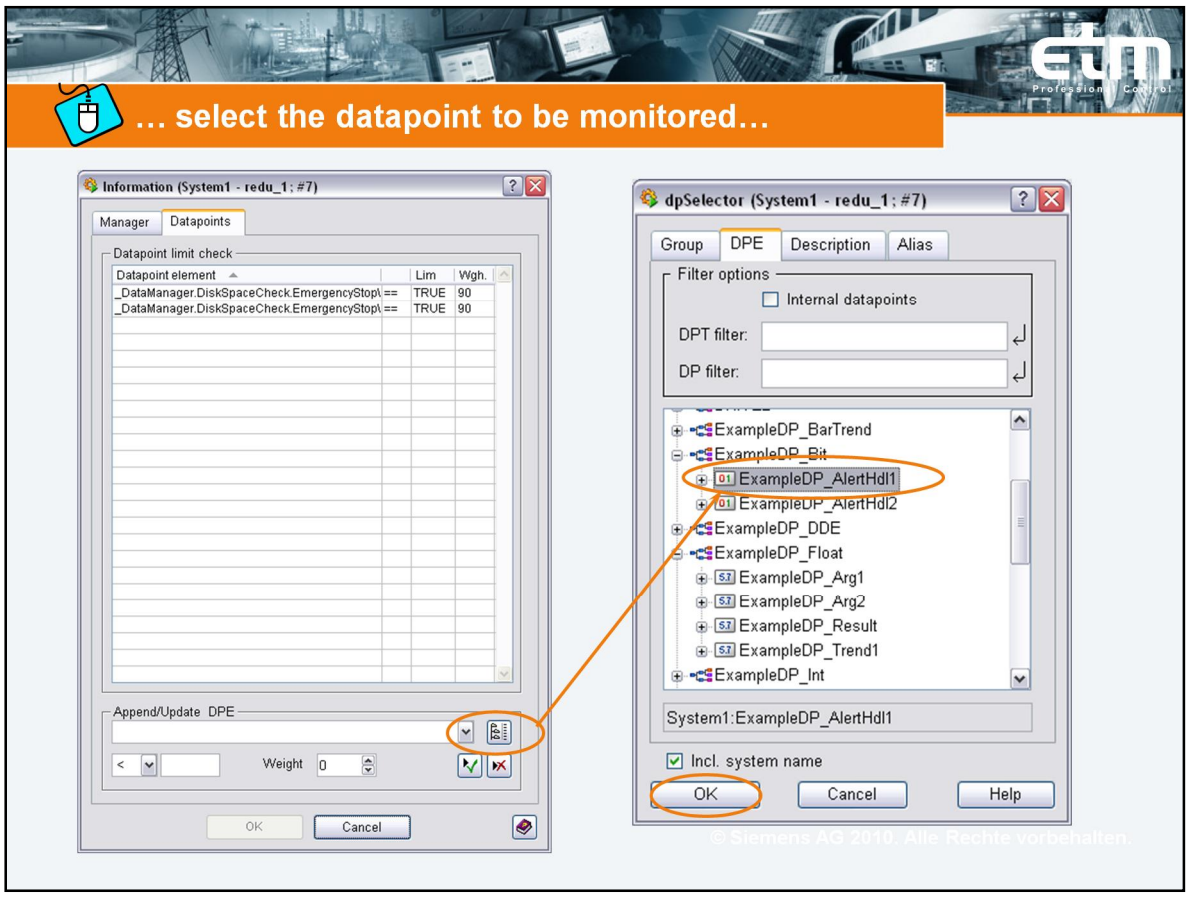
Note:

The **Error state** text field shows the following:

„sum of all **current** error-weights / sum of all **possible** error-weights"

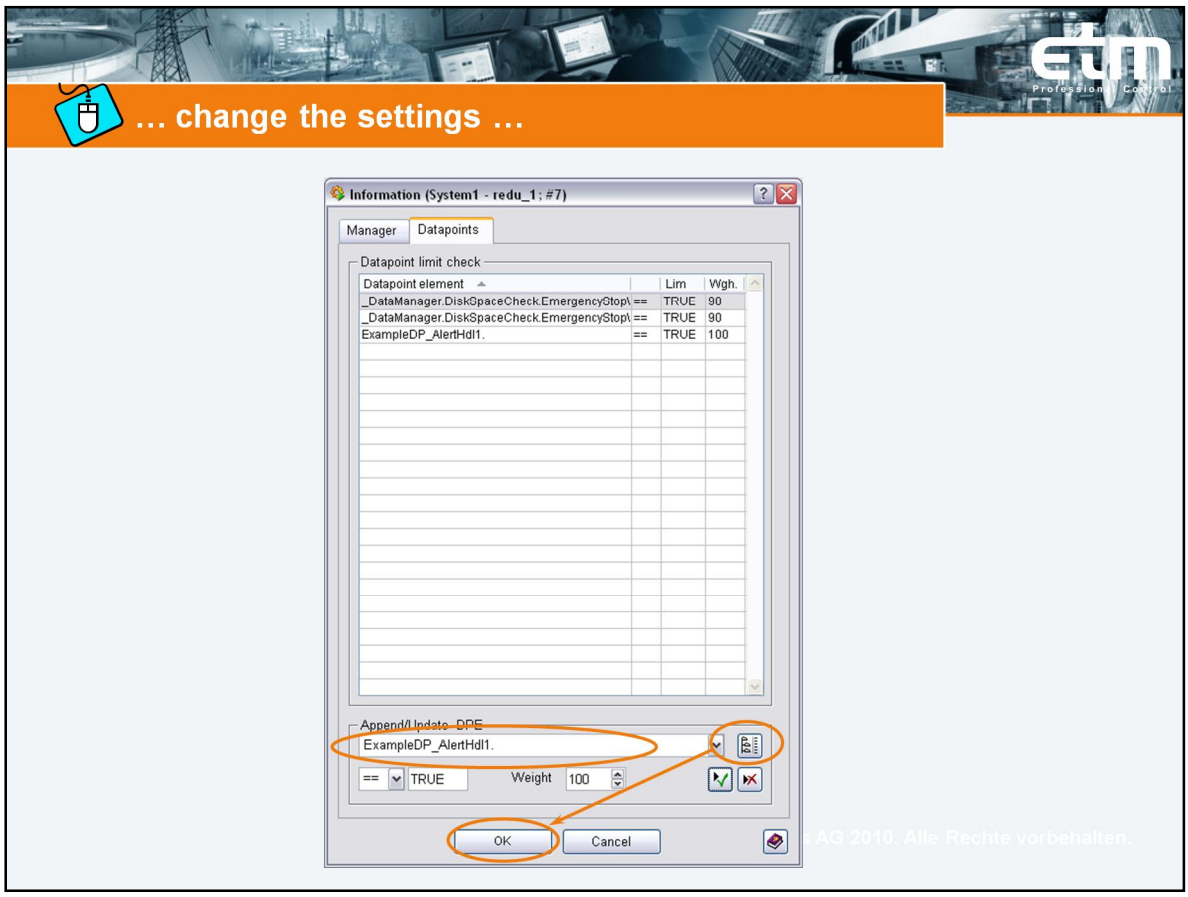
Notice that the numbers have changed from "`1/220`" to "`1/240`" .


Open the parameterization …
PVSS: System overview (System1 - redu_1; #7)
Information (System1 - redu_1; #7)
Manager
Datapoints
Datapoint limit check
Datapoint element
Lim
Wgh.
_DataManager.DiskSpaceCheck.EmergencyStop\ ==
TRUE
90
Error state
1 / 240
Userinterface #1
Preferred
Force active
Redundancy
Split mode
<host #1>
69.0%
77.0%
DM
EV
Archive
CTRL and other
Driver
ValueArchive_000
archiv_client.ctl, cal
SIM
01) 5 minutes arc
02) Hour archive
03) Day archive
04) Command arc
05) State archive
Append/Update DPE
Weight
OK
Cancel
Current user:
root
Friday 07.12.2007
3:28:56 PM



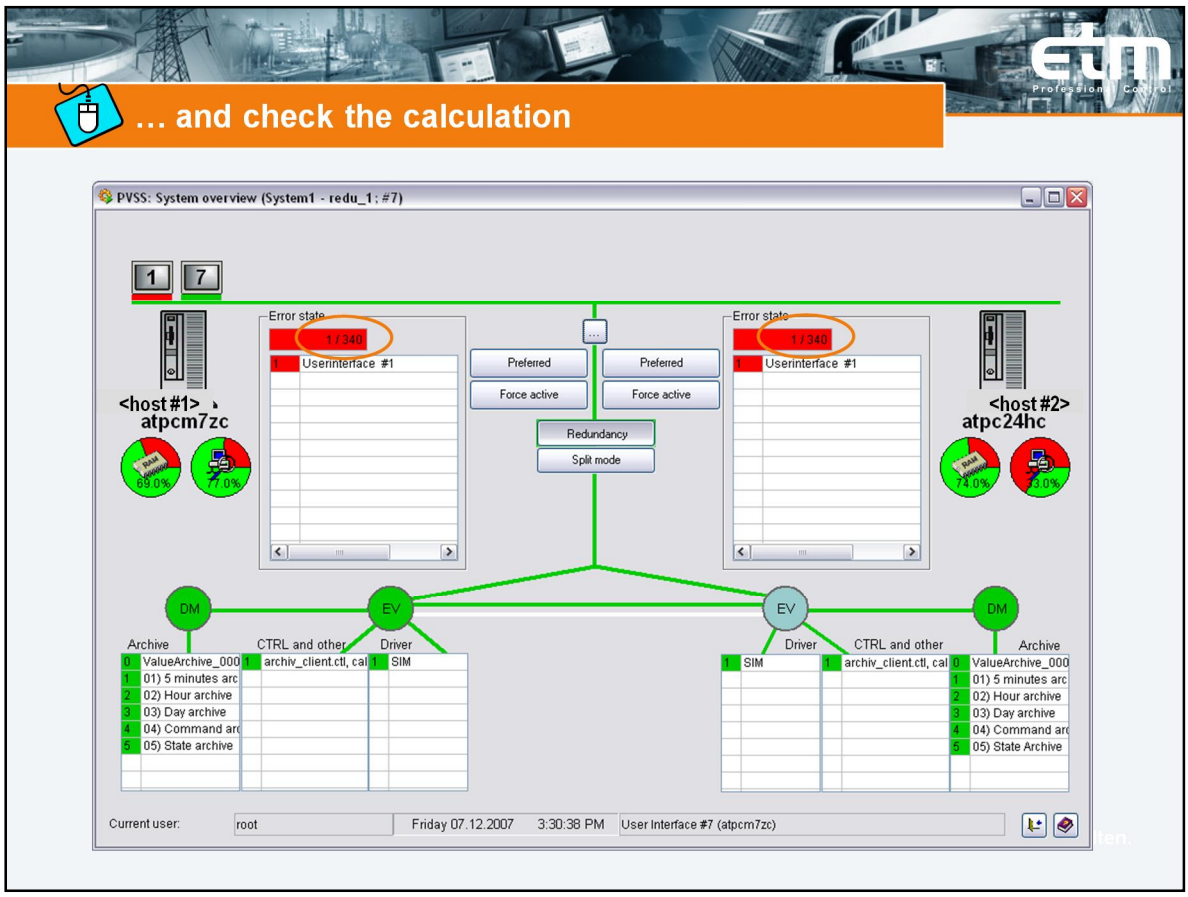
Note:

"ExampleDP_AlertHdl1" from dat point type "ExampleDP_Bit"



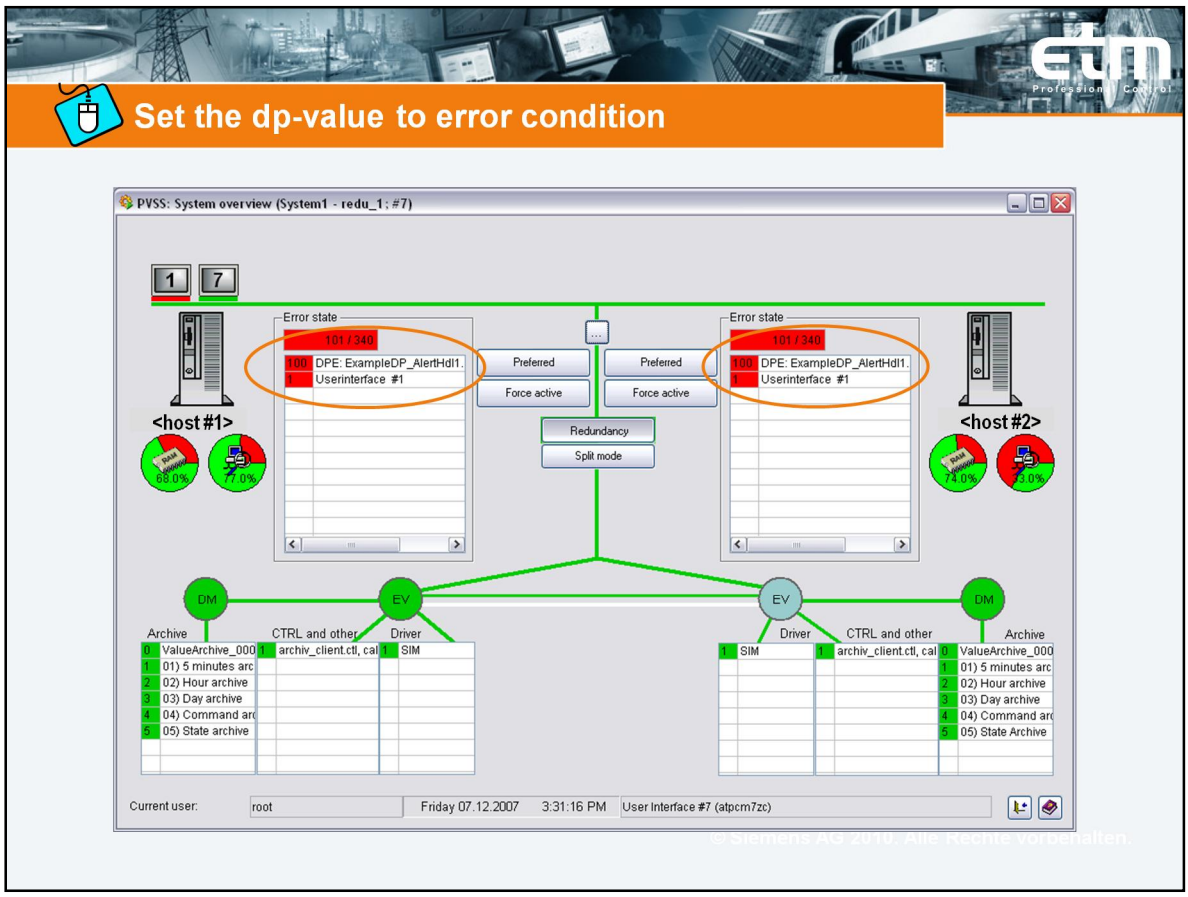
Note:

Whenever the value of this data point will be equal to TRUE, then this will contribute a weighting of 100 to the calculation of the current error state.



Note:

Notice that the numbers for „sum of all **current** error-weights / sum of all **possible** error-weights" have now changed from "`1/240`" to "`1/340`" .



Note:

The error state has now changed from " `1/340` " to " `101/340` " and the table below, that shows the detailed information about the actual error state, is also updated.

Because the value was set with the PARA module, the value has changed on the active system but of course also on the passive one. That is why there is no difference in the error state between the two systems and there is also no redu-switch.

Hints to the usage of the `config.redu`-file:

On the one hand inside the `config.redu`-file can be found common Information regarding some redundancy settings that are valid for all WinCC OA projects (i.e. how long should the active EV keep buffering in case of data recovery, etc.).

On the other hand the file may also contain some project specific Information like the settings about the different available drivers that may, of course, not be used all together at every WinCC OA project, but only the one that are really needed.

There is a already predefined `config.redu`-file available, which can be found at `<WinCC OAPath>/config`. Therein included are already all necessary common redu entries and also some project specific examples, some of them may be commented out for later use or they may be marked up in a special way.

The config.redu is, likewise to the normal config file, also divided up into several sections. There will be a [general] - section, an [event] – and a data – section. But unlike at a normal config file a certain section may show up more than one time at the config.redu. This is because the information is grouped together thematically (e.g. the section for the redu manager and also the section for the OPC driver will contain an [event] section.....

Note:

Hints to the hierarchical usage of config-files:

When files are parsed or loaded in WinCC OA, this is every time done in a hierarchical way. WinCC OA starts the search at the local project path, then continues with some optional sub-project paths and finally also the installation path.

There are also entries in the *config.redu* file located in *<WinCC OA_path>/config/*. Use the file with the same name in the project directory for project specific settings. This file is automatically read after project config file, config.level and config.[plattfrom] files. The config.level, config.platform and config.redu files are first read from the version directory, from the sub project directory and thereafter from the project directory.

Unlike the mechanism used for panels or ctrl-libraries where the file from the project directory will replace the same file from the installation folder, a config.level or also a config.redu file in the project directory does not replace a config.level or config.redu file in the version directory but the files are merged.

Hints to the Timeout settings:

All essential entries for the redundancy are predefined by default, also the timeouts. Nevertheless it may be necessary for some project to introduce specific modifications for example: If the maximum time that may be used for the initial recovery of the database in case of a redundant system is reached, an error message is shown and the recovery will be aborted. The maximum time that may be used for the copying may be defined by using the entry passiveRecoveryTimeout, e.g. passiveRecoveryTimeOut = 10.

Hints to the usage of the fwdDp keyword:

With the keyword fwdDp (= forward data points) the changes of the original attributes of the specified DP element are forwarded from the Event manager automatically to the redundant Event manager (regardless of whether the value is set on the active or passive Event). If the element is a node this applies for all leaves under the node. In order to pass different elements this keyword can be specified several times. Via passing the elements changes that can only occur on one of the computers (e.g. disk full, connection status to periphery) can be sent to the other computer in a redundant system.

Hints to the usage of the fwdDpType keyword:

With the key word "fwdDpType" (like fwdDp) all changes on this DP element are passed from the Event Manager automatically to the redundant Event Manager. Contrary to “fwdDp" a DP element of an existing data point is not specified but the element of a DP type in the form "Type.Element" (e.g.. "ExampleDP_Bit." for the root element of the DP type "ExampleDp_Bit"). Thus, the corresponding elements of all data points of this type are forwarded.

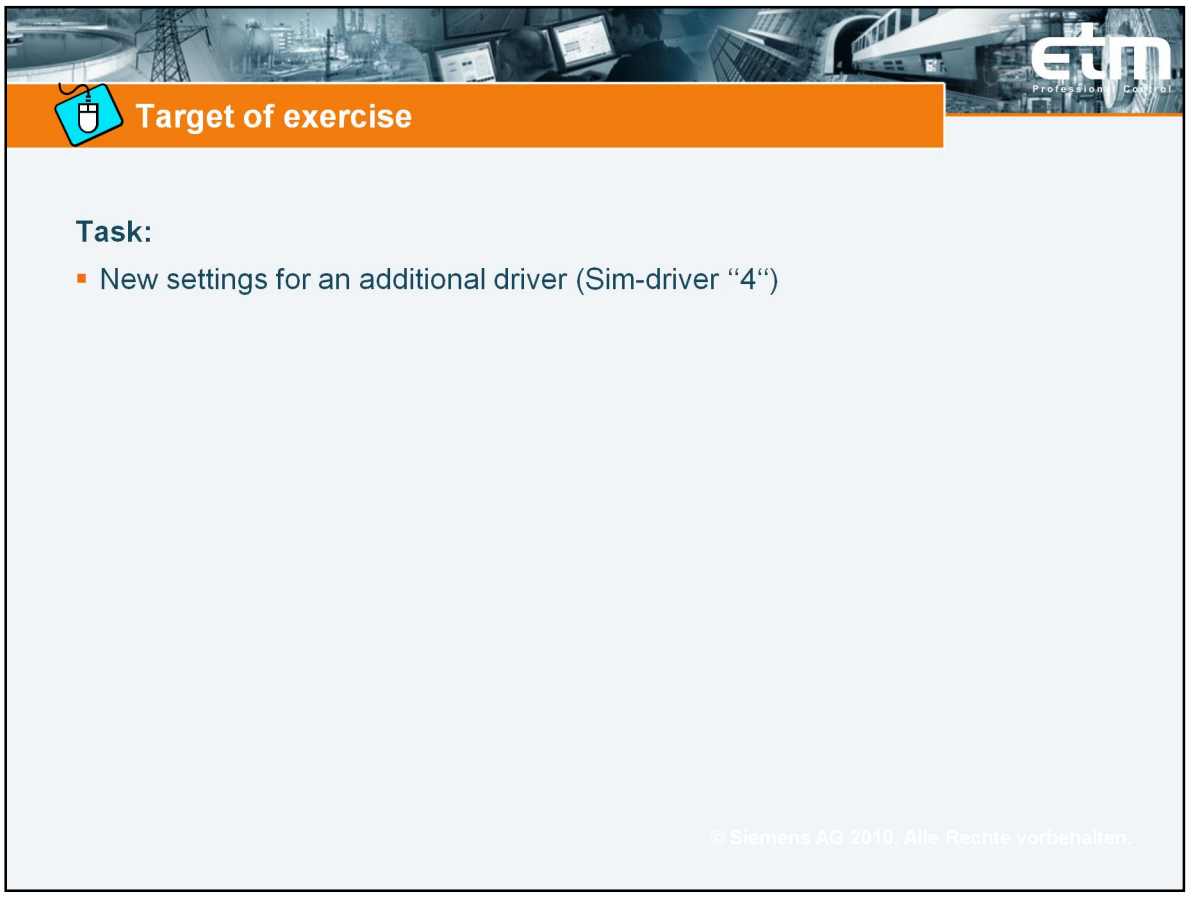
Hints to the usage of the copyDp keyword:

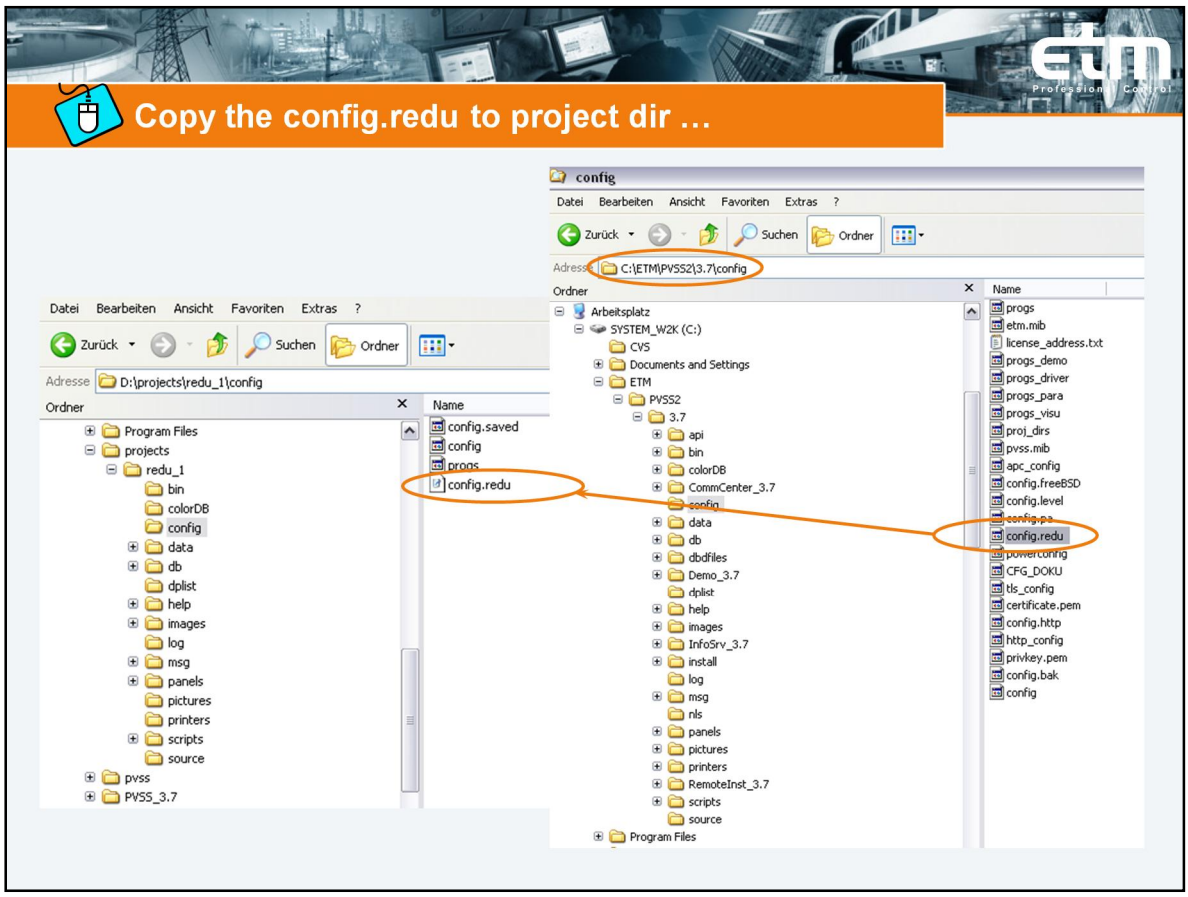
Via the keyword copyDp (copy data points) the changes of the original attributes of the specified source DP element are also made on the same element of the specified target data point. The target data point has to be of the same type as the source data point. If the element is a node this applies for all leaves under the node. In order to duplicate different elements the keyword can also be specified several times. Copy DPs are elements that normally are equal on both computers like e.g. parameterizations or triggering of commands (GA, change of the archive set).

Hints to the usage of the copyDpType keyword:

Has the same behaivour like copyDP, but here already on DP-Type level. Automatically all existing DPs from the specified DP-Type get copied.

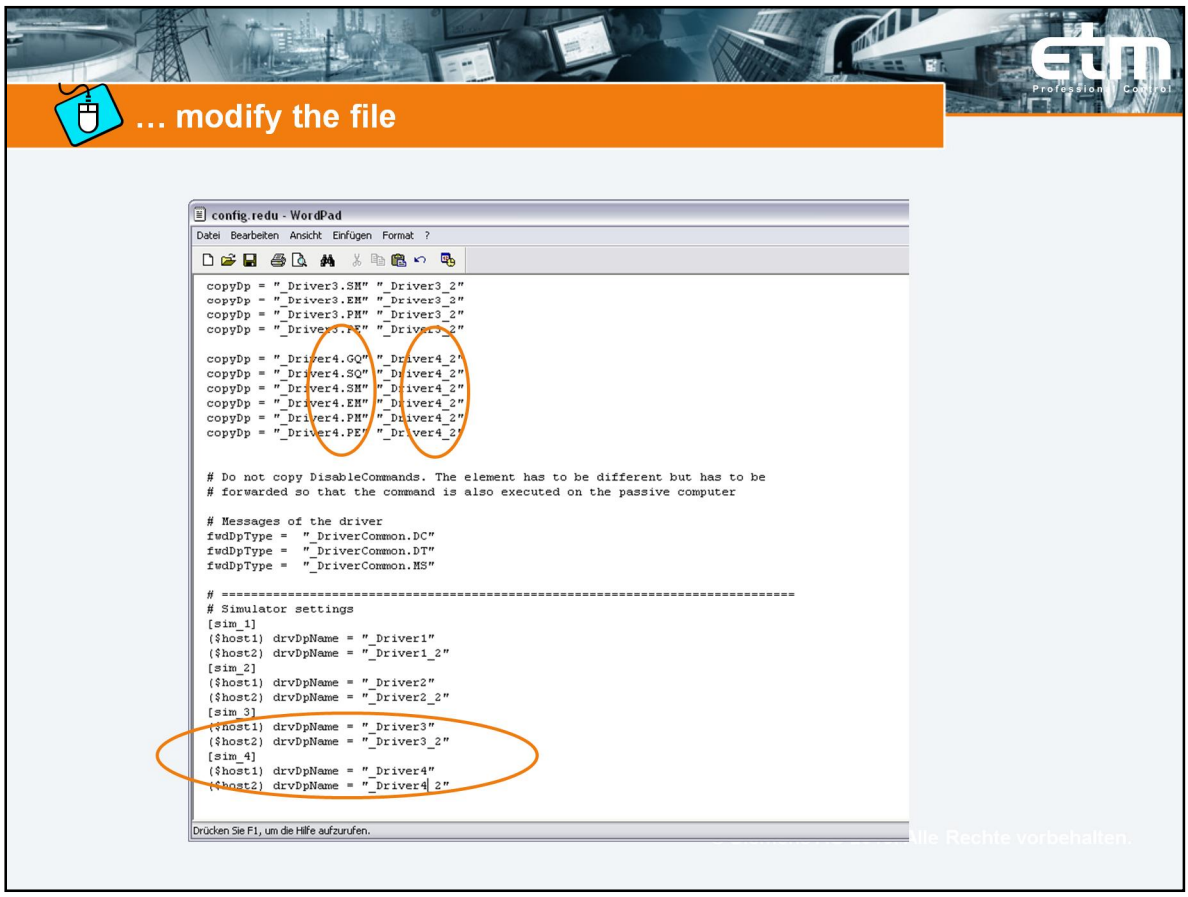
That‘s specially necessary in case of there are exisiting a dynamic number of DPs in a system e.g. Connection Datapoints from Drivers (each per PLC). It‘s not necessary to modify the config.redu file, if more devices (PLC‘s) are created.


Target of exercise
Task:
New settings for an additional driver (Sim-driver “4“)
© Siemens AG 2010. Alle Rechte vorbehalten.



Note:

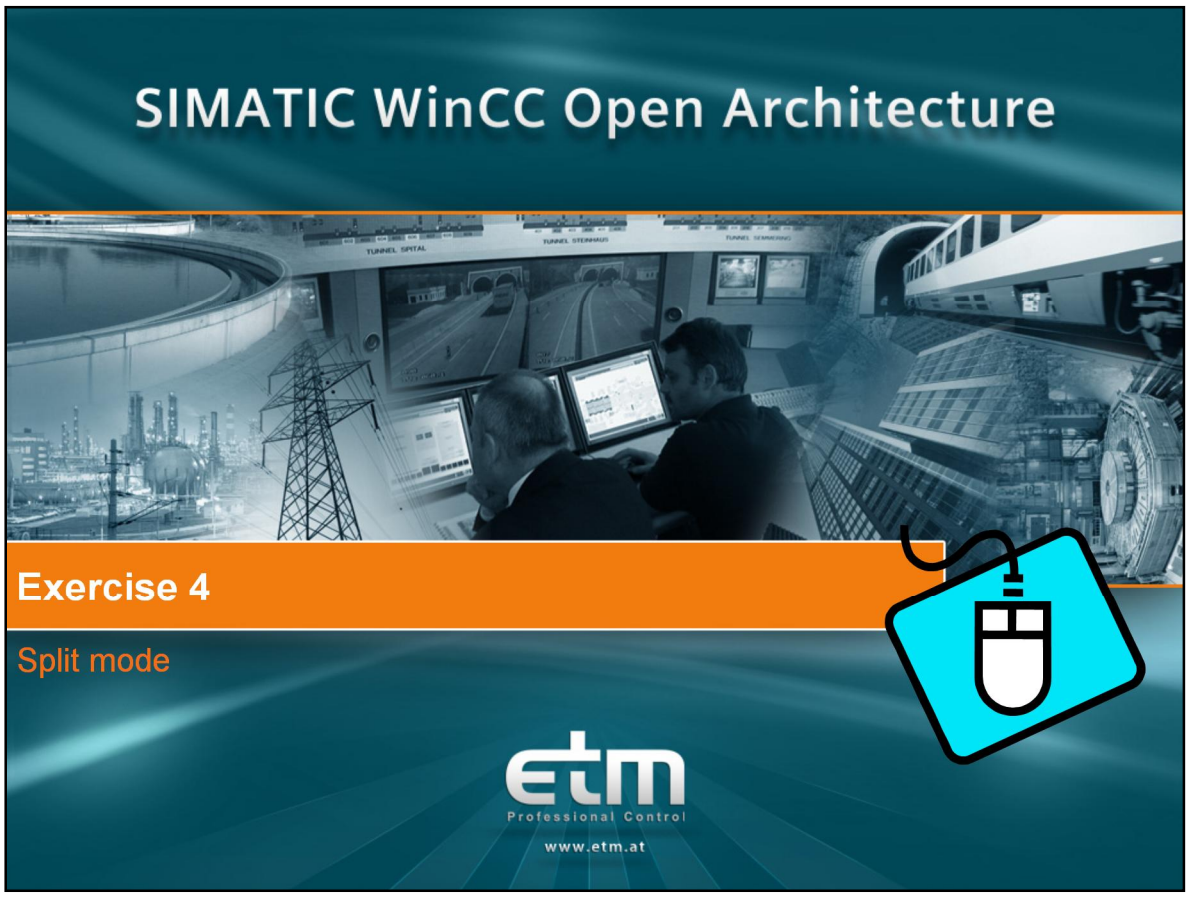
The config.redu-file should be copied from the WinCC OA installation folder to the project folder first, before starting the adaptation of the file.

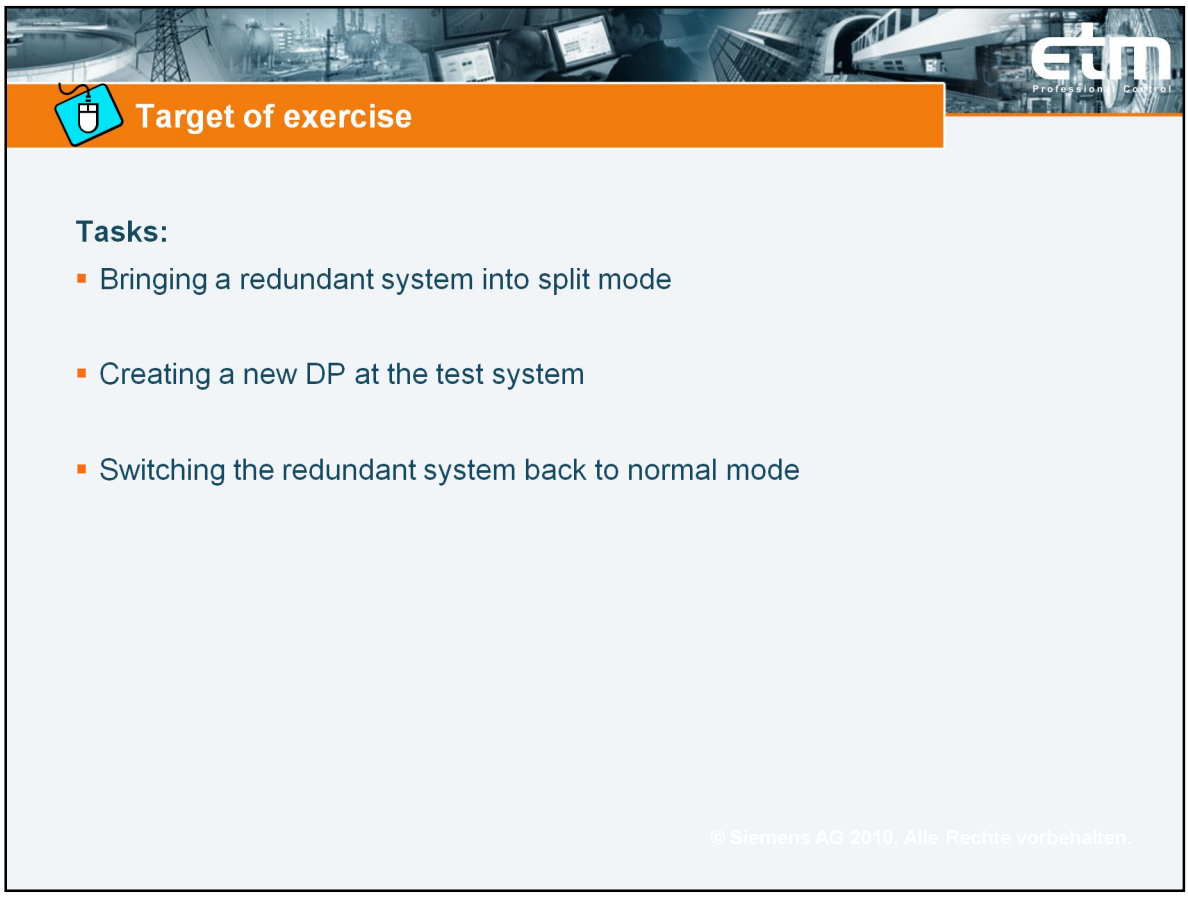


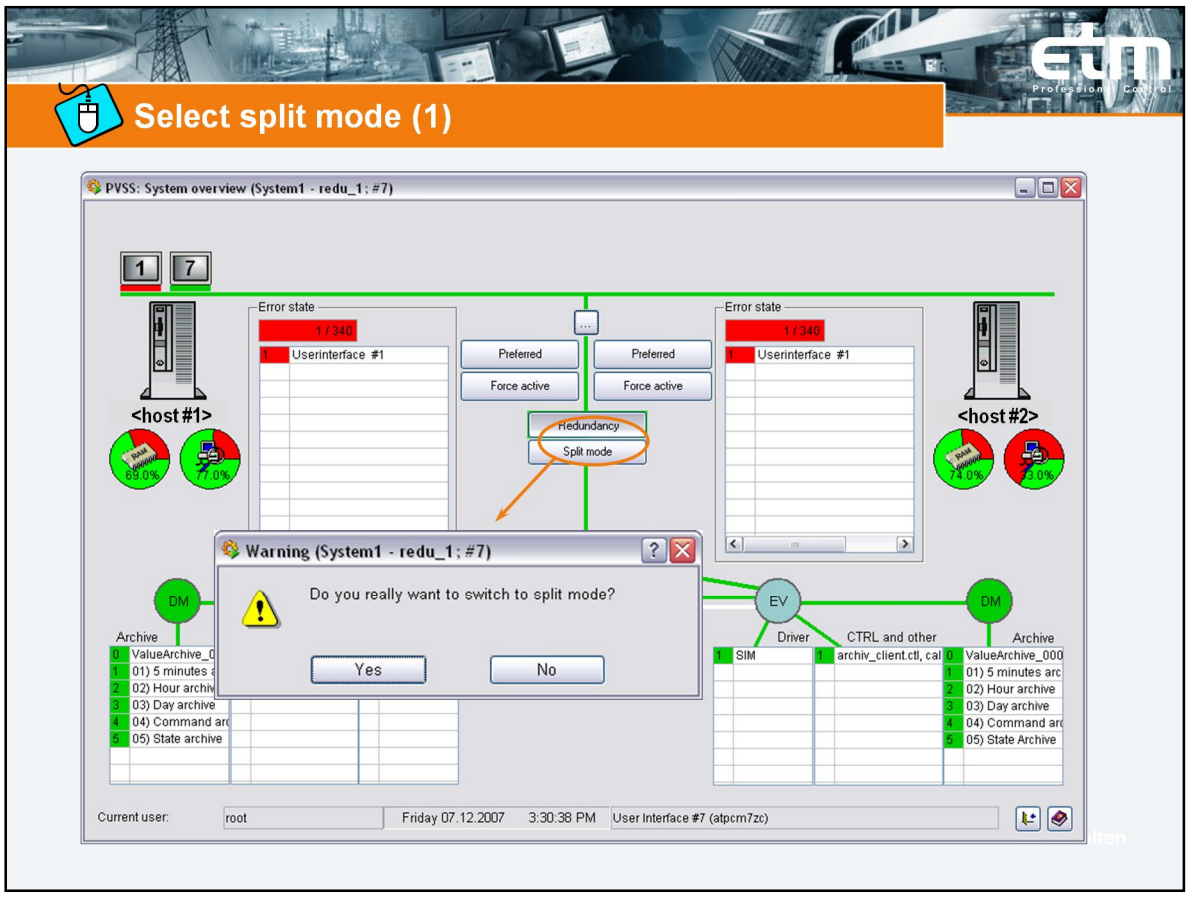
Note:

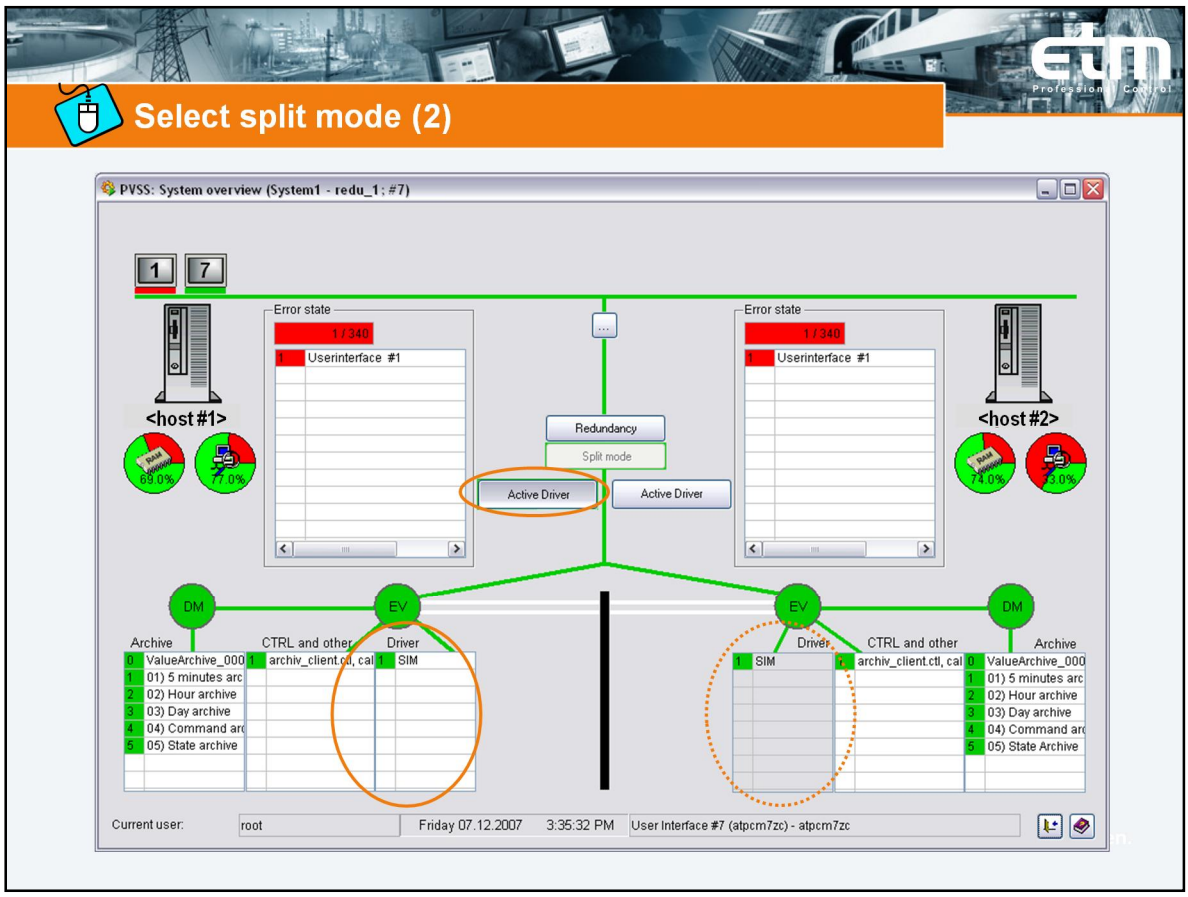
After the `config.redu`-file was adapted, the data points for driver number 4 might be added to database if this hasn´t been done yet (`_Driver4` und `_Driver4_2` vom Typ `_DriverCommon`).

Keep in mind to copy the config.redu-file also to the other system and then both systems have to be restarted that the changing my take effect (config file entries will be read at the startup of the managers).


SIMATIC WinCC Open Architecture
Exercise 4
Split mode
etm
Professional Control
www.etm.at


Target of exercise
Tasks:
Bringing a redundant system into split mode
Creating a new DP at the test system
Switching the redundant system back to normal mode
© Siemens AG 2010. Alle Rechte vorbehalten.


Select split mode (1)
PVSS: System overview (System1 - redu_1; #7)
1
7
Error state
1 / 340
Userinterface #1
Preferred
Force active
Preferred
Force active
Redundancy
Split mode
Error state
1 / 340
Userinterface #1
<host #1>
<host #2>
Warning (System1 - redu_1; #7)
Do you really want to switch to split mode?
Yes
No
DM
Archive
0 ValueArchive_0
1 01) 5 minutes
2 02) Hour archiv
3 03) Day archive
4 04) Command ar
5 05) State archive
EV
Driver
1 SIM
CTRL and other
1 archiv_client.ctl, cal
DM
Archive
0 ValueArchive_000
1 01) 5 minutes arc
2 02) Hour archive
3 03) Day archive
4 04) Command ar
5 05) State Archive
Current user:
root
Friday 07.12.2007 3:30:38 PM
User Interface #7 (atpcm7zc)



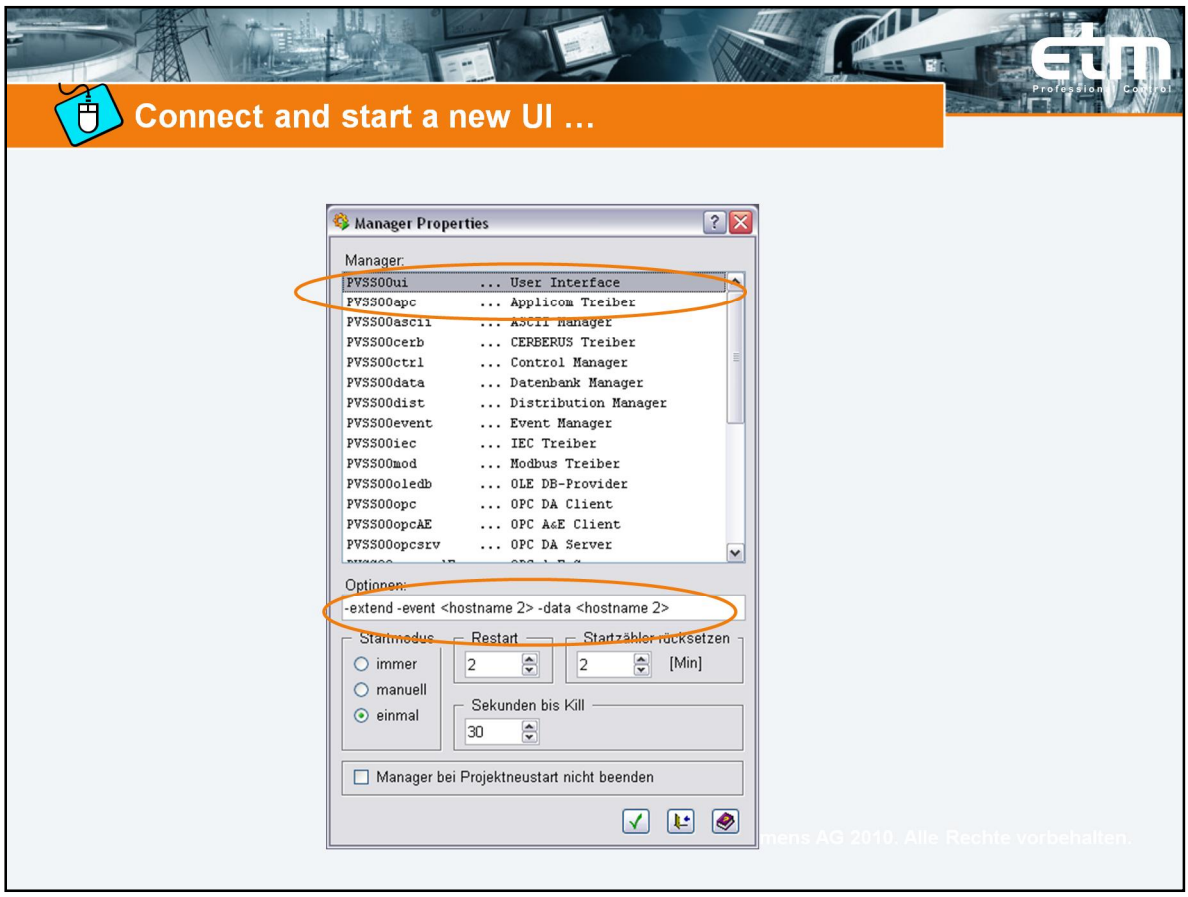
Note:

The system that was active when going into split mode will keep the active driver connection to the peripherals, the passive system will become the testing system.

By clicking one of the **Active Driver** buttons all of the driver connections will be switched from one to the other system. An active driver is capable of sending and receiving messages from the peripheral units, a passive driver only receives messages, but sends out nothing.

With the help of the split manager the systems are exchanging internal data (connection state, memory usage, etc.). Should there be the need of exchanging additional data during split mode, the predefined datapoint-groups `SplitGet`, `SplitGet_2`, `SplitConnect` and `SplitConnect_2` may be adjusted accordingly.

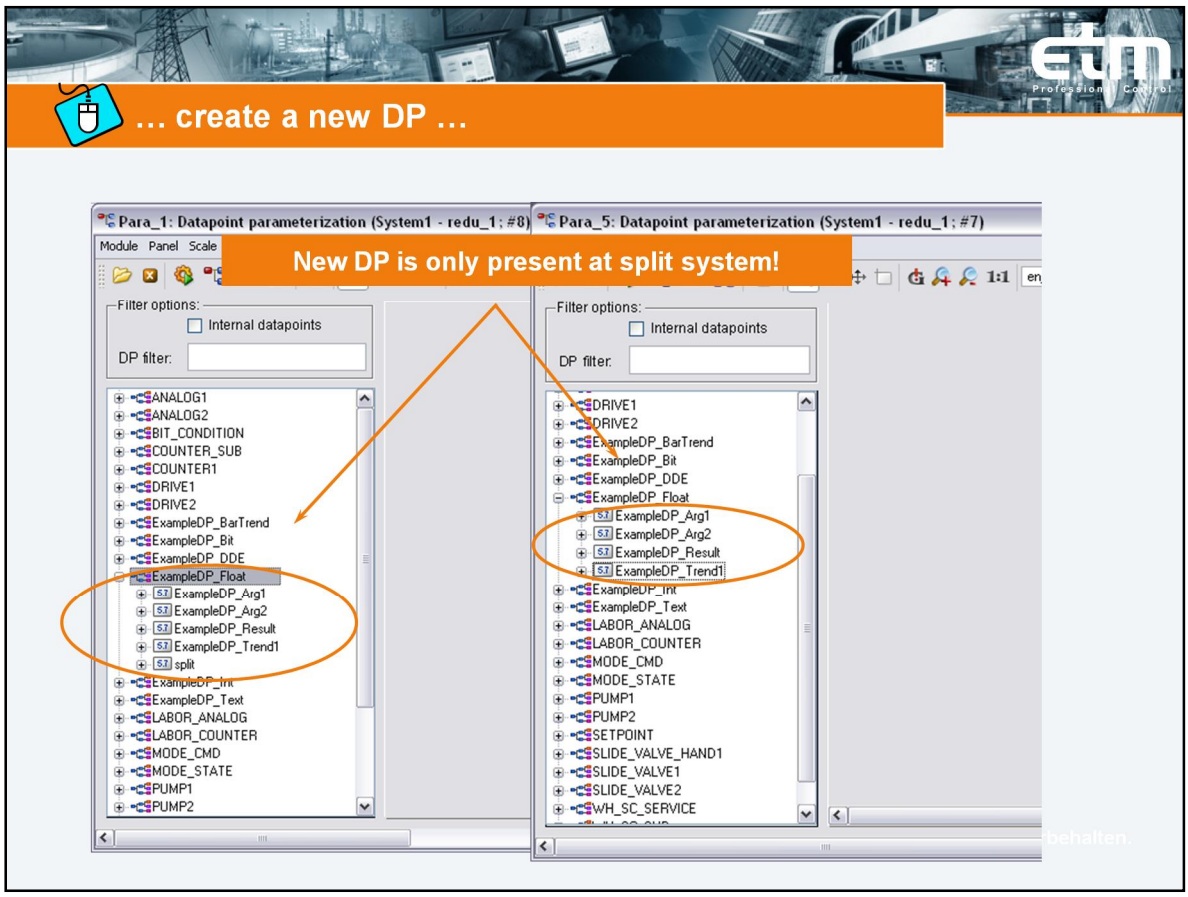
As long as the system remains in split mode (split-manager is active), the redu-manager is sent to stand-by- mode.



Note:

At one of the two hosts a new UI is started in PARA mode, that will only connect to the testing system. This is done by stating the data- and the event host name to which it shall connect as an option for the startup command of this UI:

```
-m para -extend –data <hostnameTestsys> -event <hostnameTestsys>
```



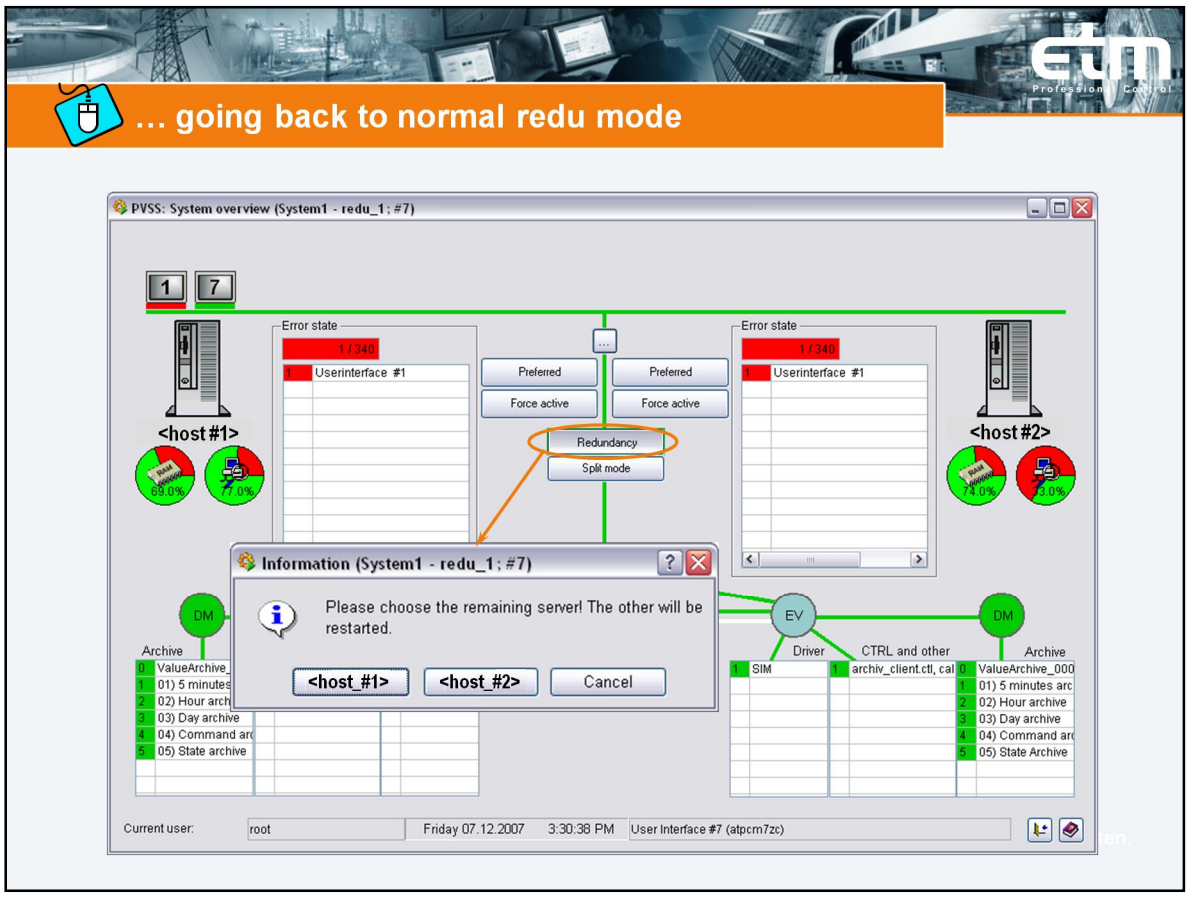
Note:

The creation of the data point named `split` from type `ExampleDP_Float` was done at a PARA module, that was only connected to the testing system.

Therefore it is clear that this data point only exists at the database of the test system but not on the other system.

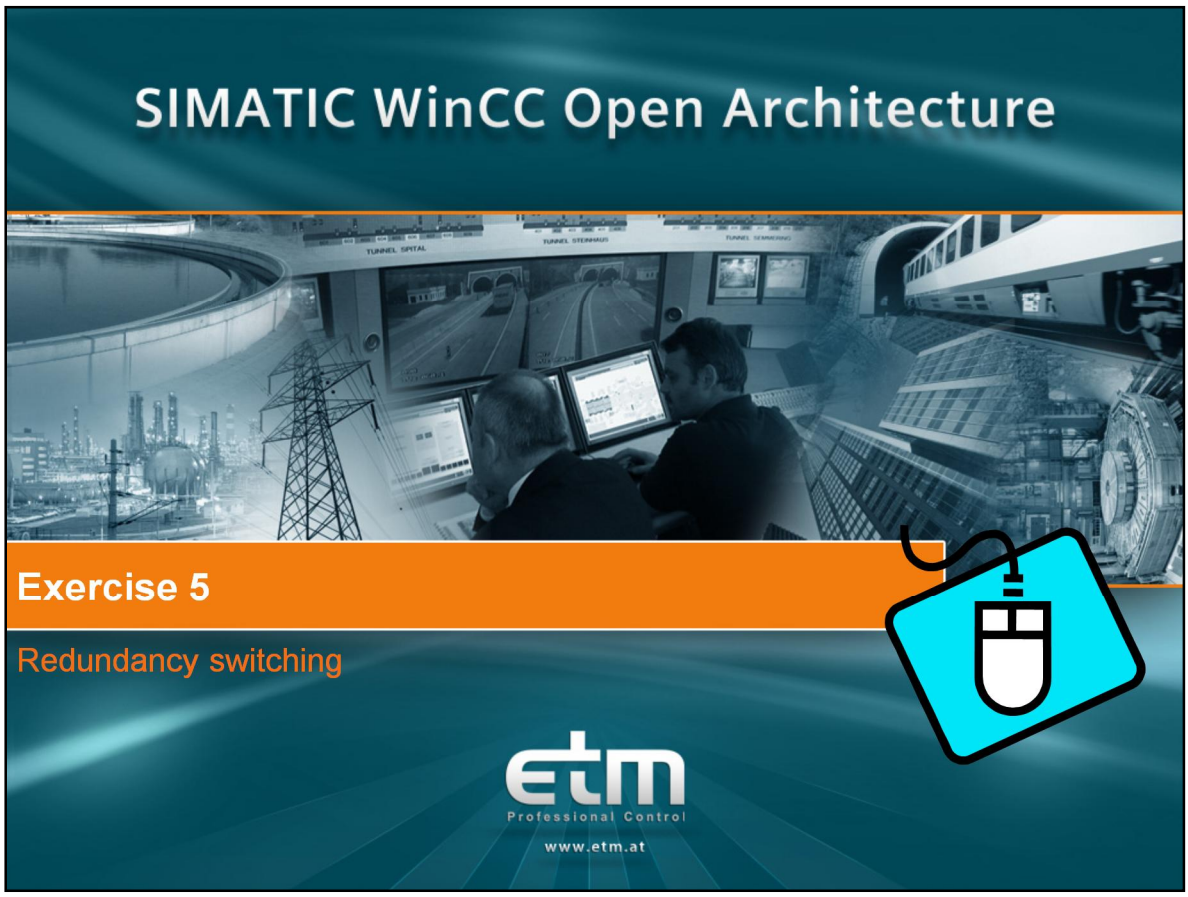
For normal a UI is not connected to only one of the two redundant hosts but to both, therefore these “normal” UIs would see both systems data rooms.

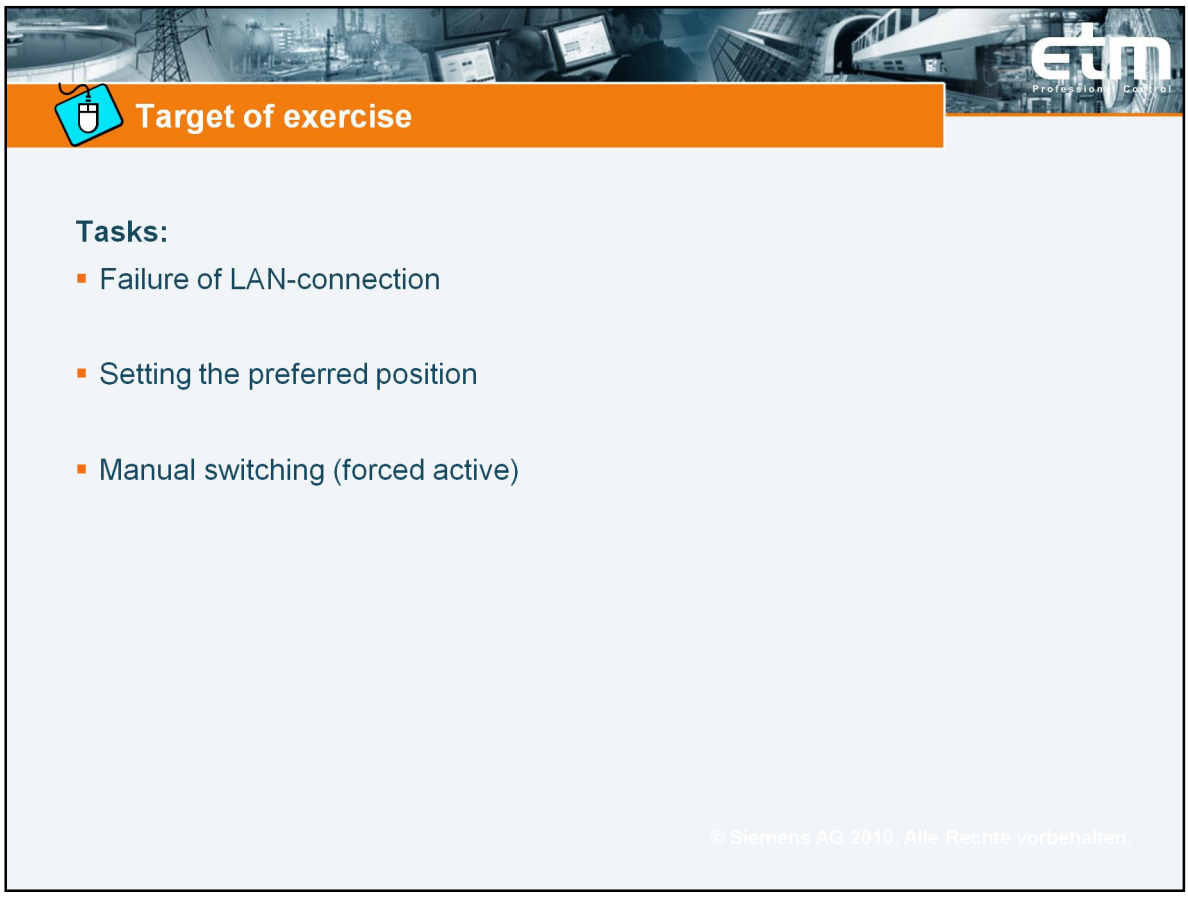
By using the `data <hostname>` and `event <hostname>` option (either in the config-file or as a startup option) a UI will only connect to the system with the given hostname (the active or the testing system) and therefore this UI will be only aware of the data existing at that systems local database.

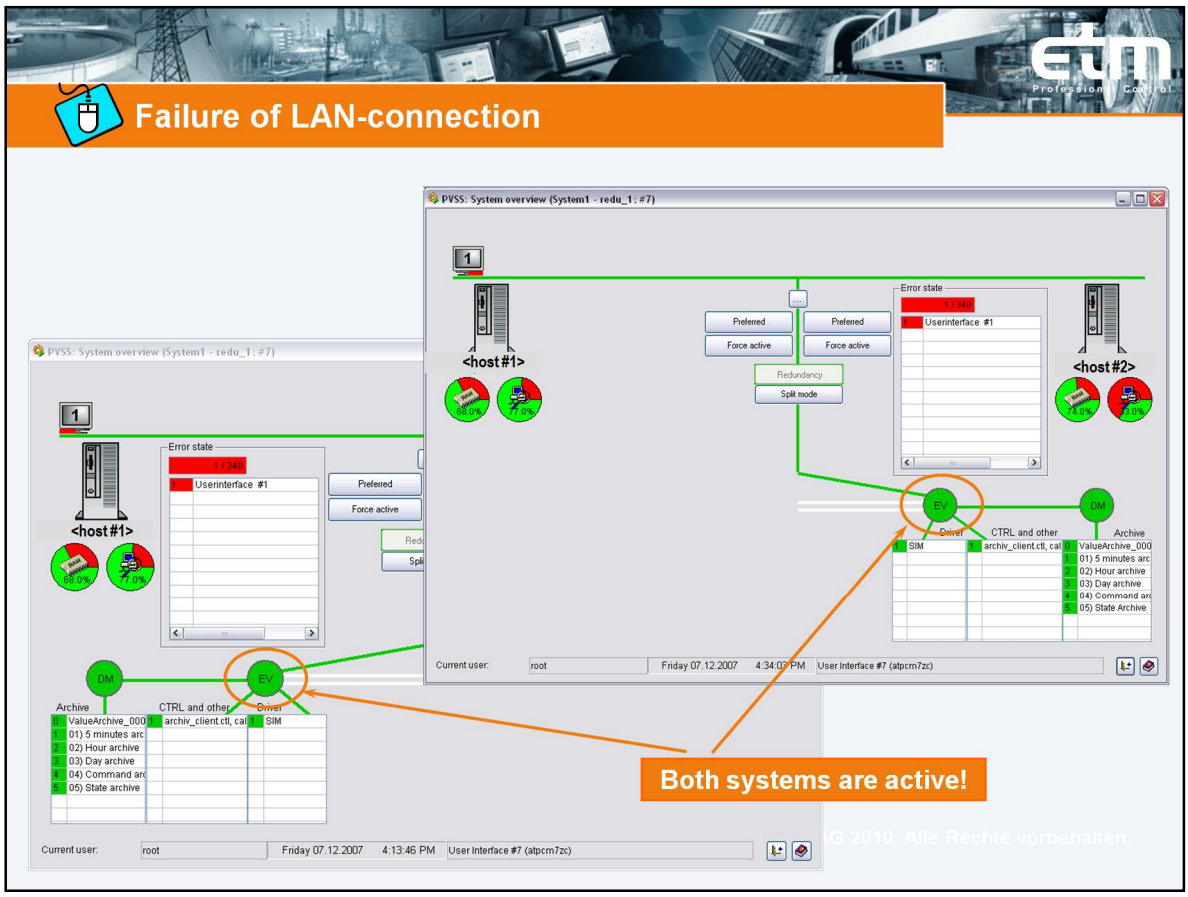


Note:

By selecting which of the two system shall remain, it is also selected which set of data should be taken back into normal operation mode. The other system is restarted because at the restart of a redundant system an automatic data alignment to the already running system is done.


SIMATIC WinCC Open Architecture
Exercise 5
Redundancy switching
etm
Professional Control
www.etm.at

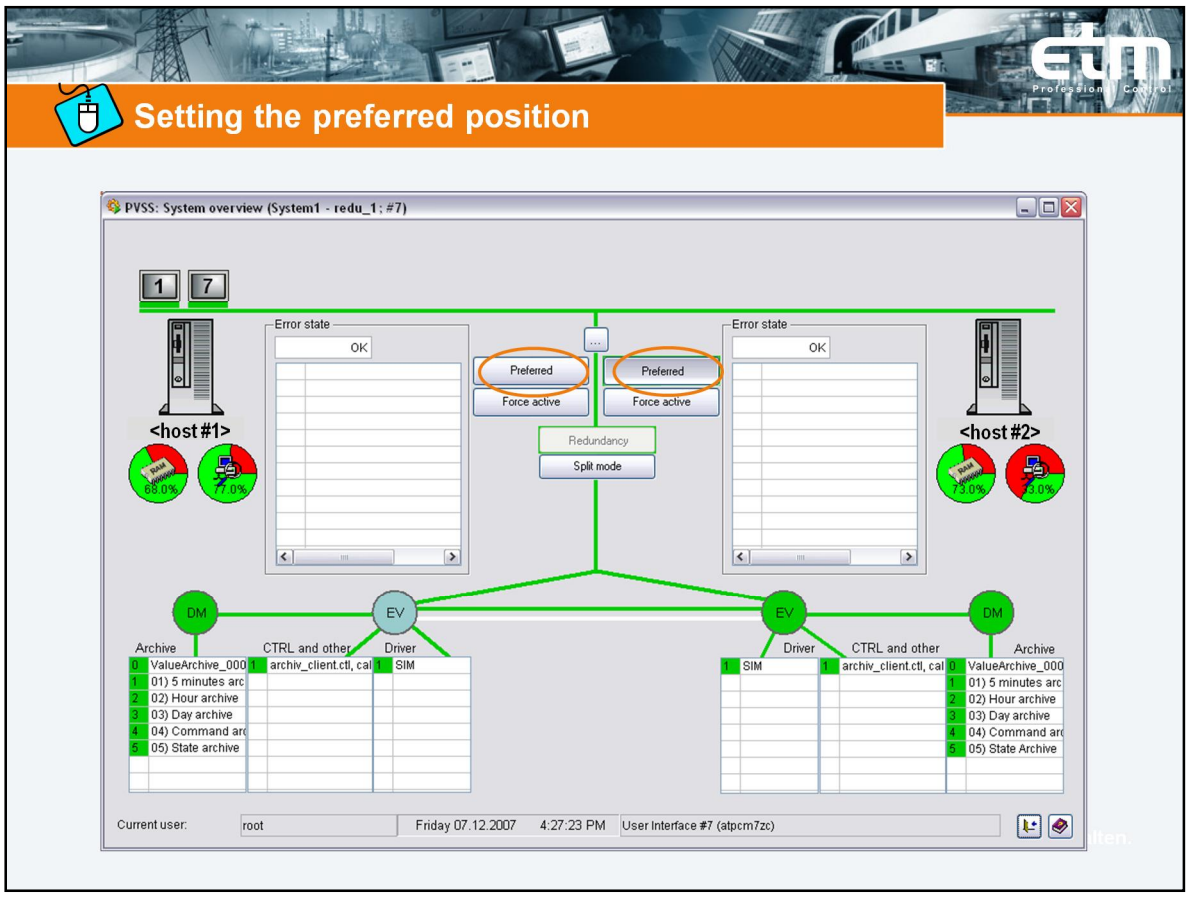

Target of exercise
Tasks:
Failure of LAN-connection
Setting the preferred position
Manual switching (forced active)
© Siemens AG 2010. Alle Rechte vorbehalten.



Note:

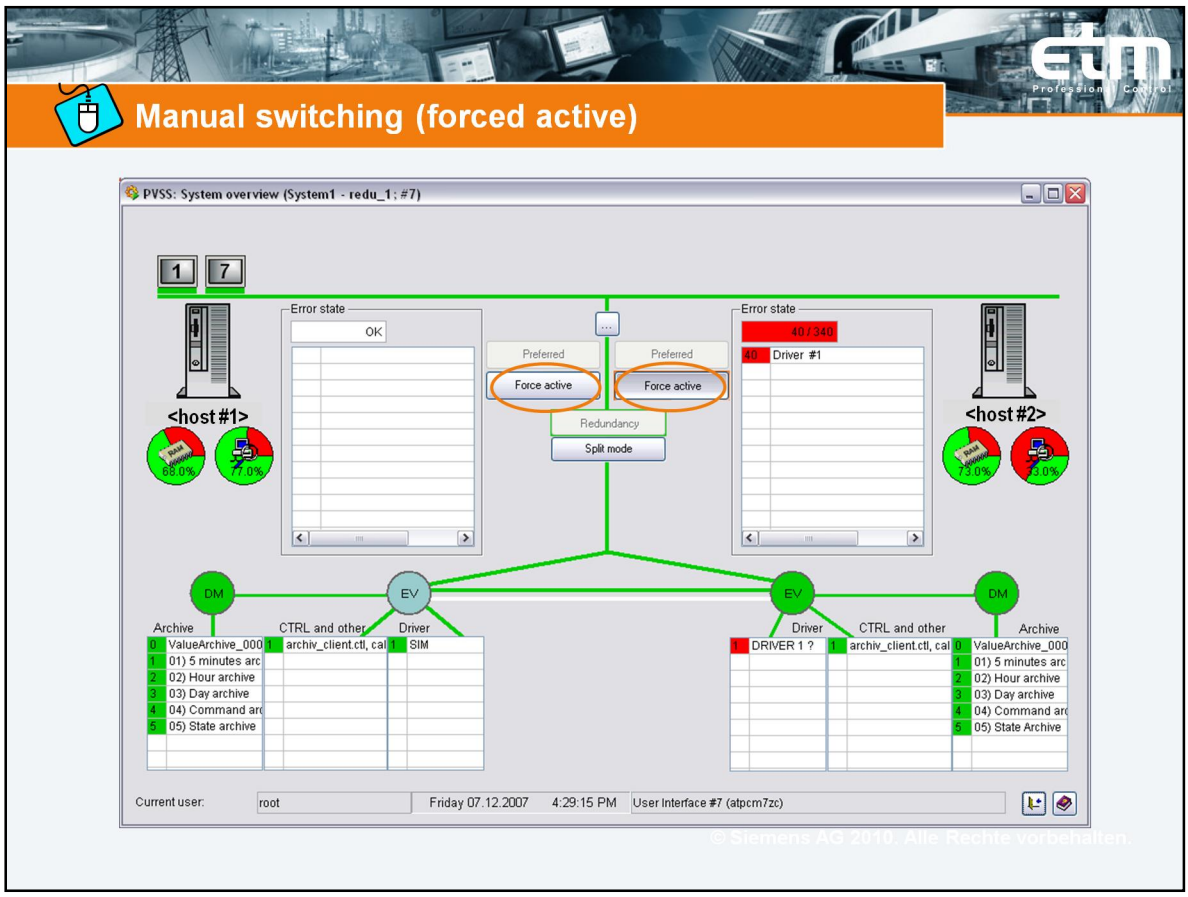
When the network connection between the two servers is lost both systems will become active because they are not getting alive packets from the partner anymore and therefore the other system will be considered lost.

When the network connection is up again, the system with the higher error state or the system that was passive before the connection failure, is being restarted, to ensure data consistency.



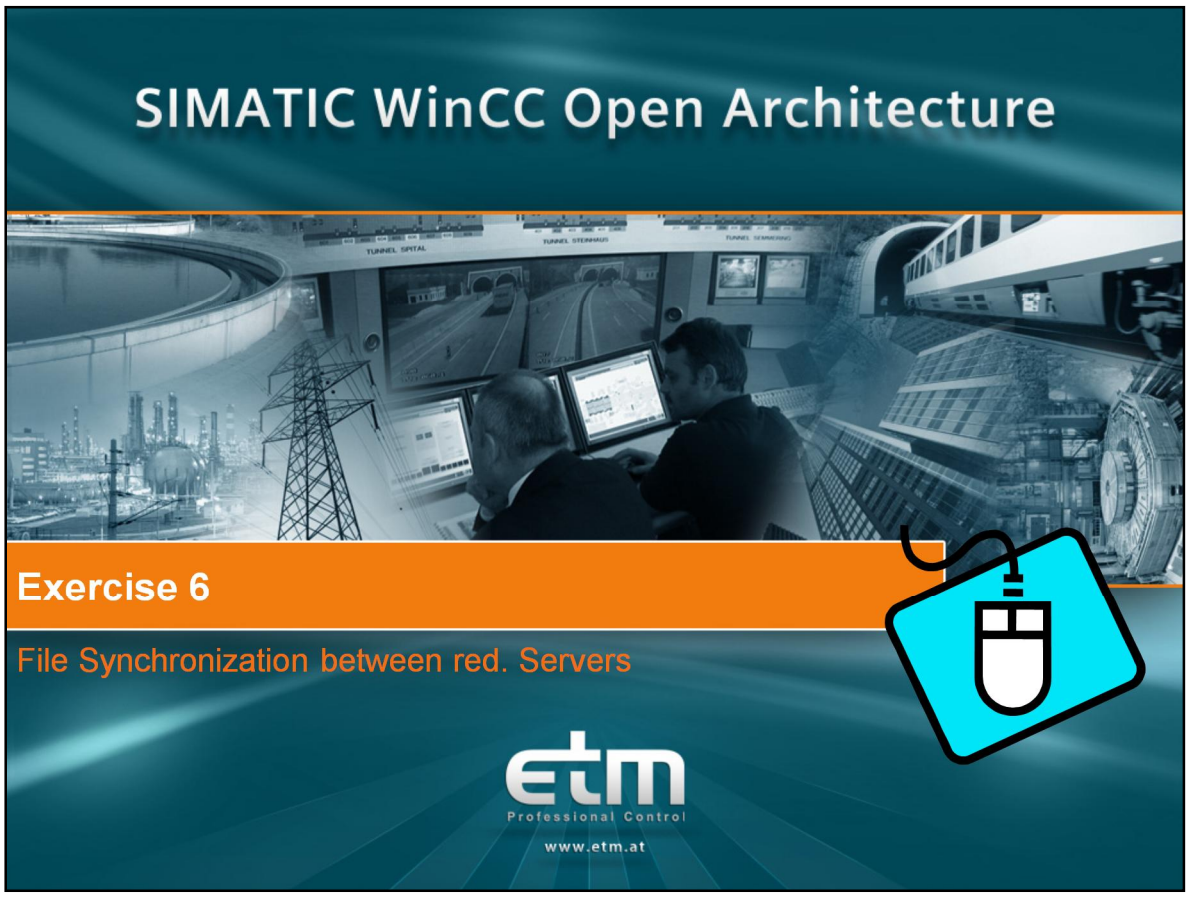
Note:

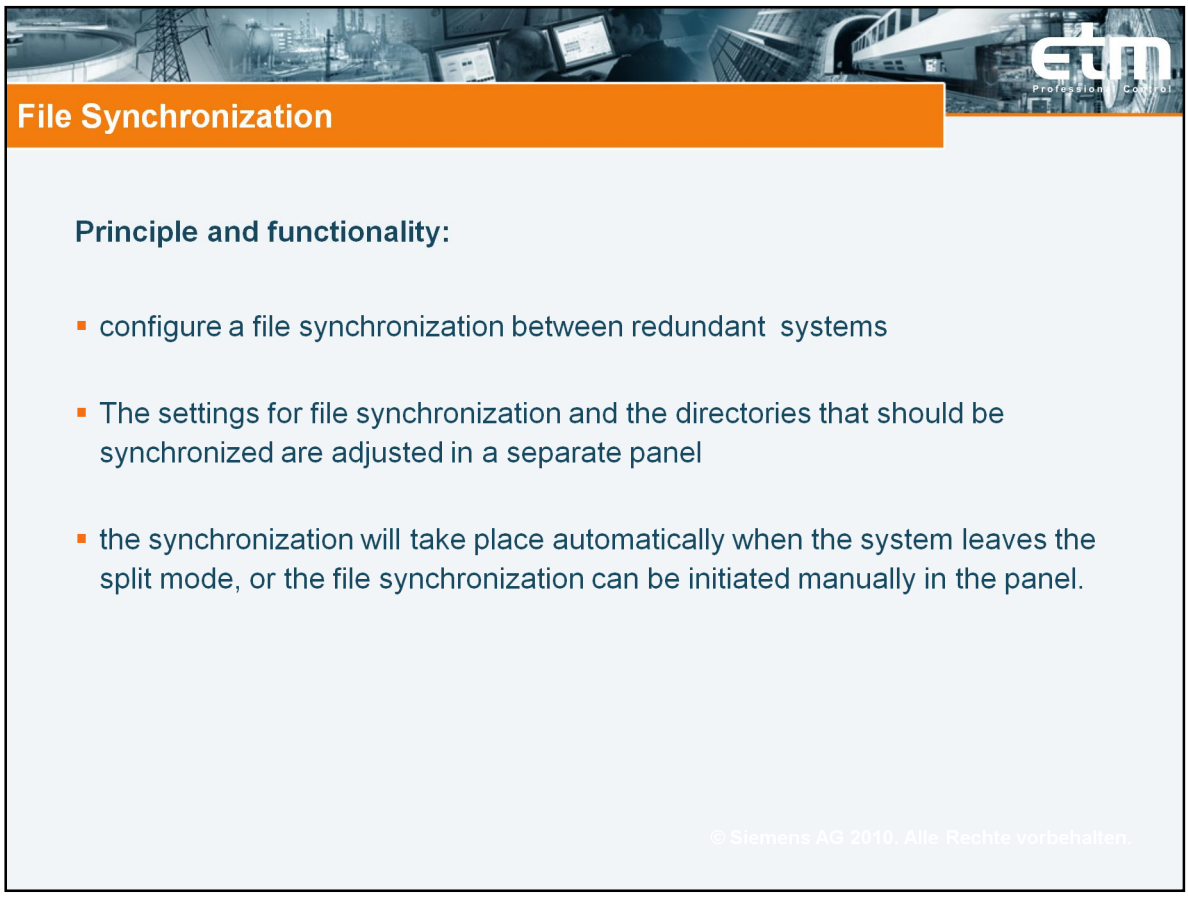
In the case of **equal error states** the active system may be selected, by using one of the **preferred** buttons.



Note:

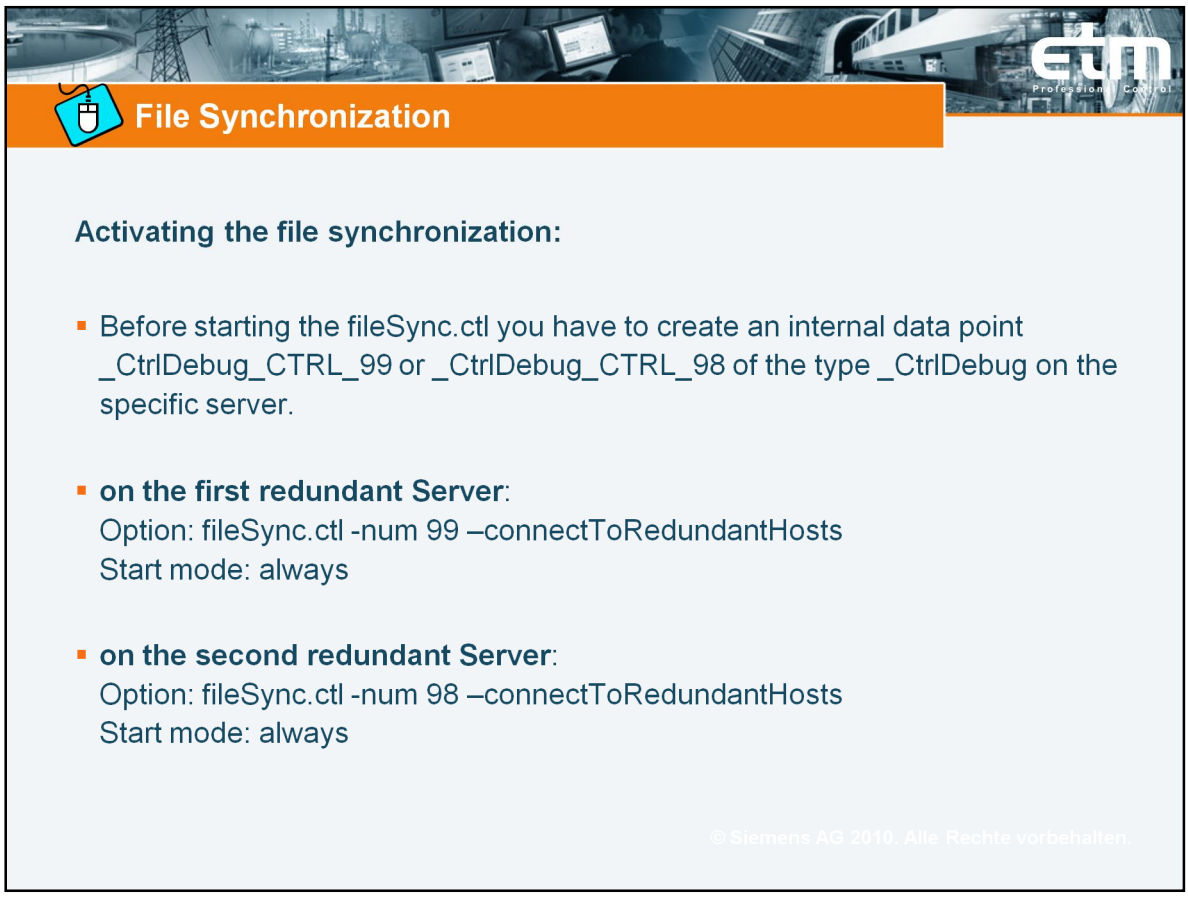
In the case of **unequal error states** the active system may be set to the right or to the left system by using one of the **force active** buttons.


SIMATIC WinCC Open Architecture
Exercise 6
File Synchronization between red. Servers
etm
Professional Control
www.etm.at



Note:

In a redundant system all raima-db data (PARA) is automatically synchronized between the red. Server pair. All configuration data files(Gedi) needs to be synchronized manually with the CTRL-Script „filesync.ctl“.



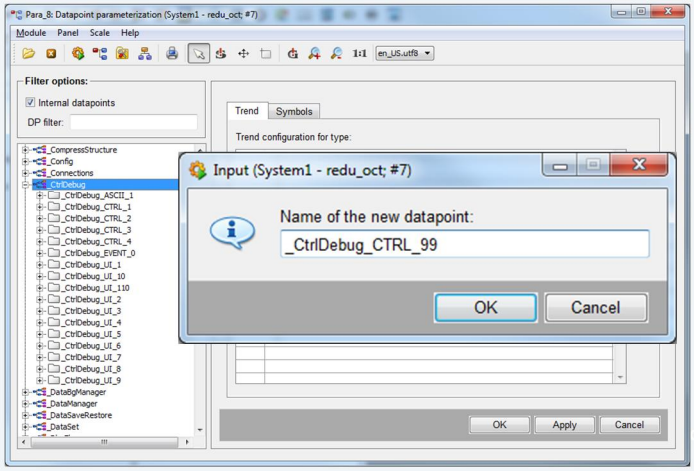
Note:

Please proceed the follwing steps to configure the manully file synchronization between the redundant server pair.

File Synchronization

Activating the file synchronization:

- Before starting the fileSync.ctl you have to create an internal data point _CtrlDebug_CTRL_99 or _CtrlDebug_CTRL_98 of the type _CtrlDebug on the specific server.



Note:

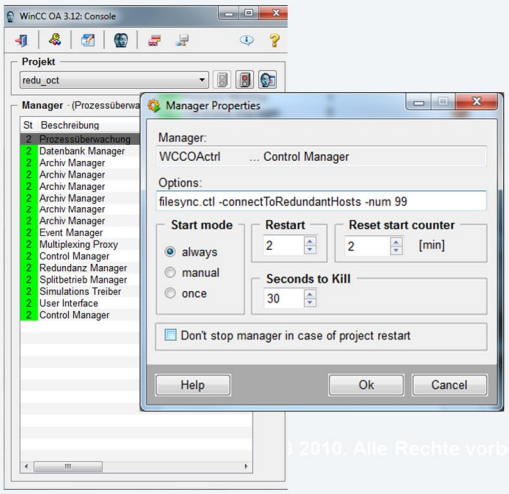
Create an internal datapoint „_CtrlDebug_CTRL_99" because of storing the DEBUG-information.

Please **create also on the second redundant Server a datapoint**, but there with the name **„_CtrlDebug_CTRL_98".**

File Synchronization

Activating the file synchronization:

- **on the first redundant Server**:
  Option: fileSync.ctl -num 99 -connectToRedundantHosts
  Start mode: always



Note:

Create on the first red. Server a CTRL-Manager with the option „fileSync.ctl -num 99 -connectToRedundantHosts“.

The manager numbers needs to be the same as the previuosly created CTRL_DEBUG datapoints.

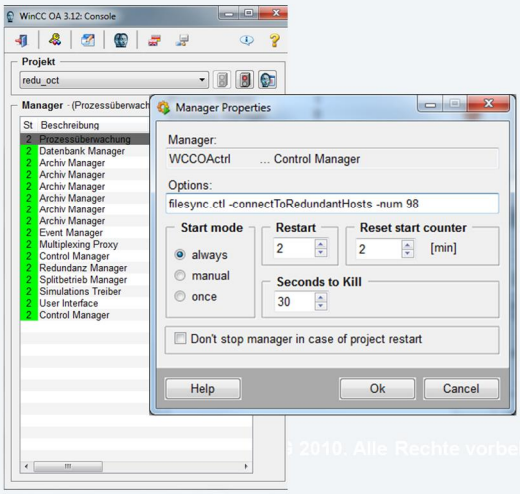
The manager numbers can be freely choosen, but be aware that they have to be different on both redundant servers.

In this example was used „-num 99“ and „-num 98“.

File Synchronization

Activating the file synchronization:

- **on the second redundant Server**:
  Option: fileSync.ctl -num 98 -connectToRedundantHosts
  Start mode: always



Note:

Create on the first red. Server a CTRL-Manager with the option „fileSync.ctl -num 99 -connectToRedundantHosts“.

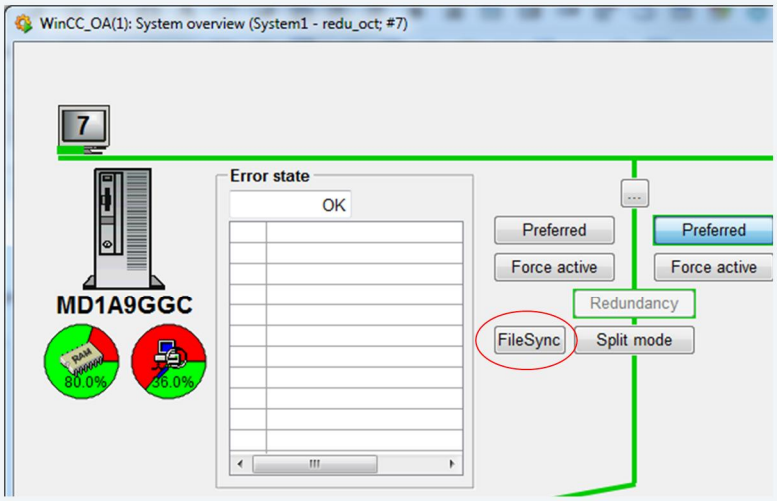
The manager numbers needs to be the same as the previuosly created CTRL_DEBUG datapoints.

The manager numbers can be freely choosen, but be aware that they have to be different on both redundant servers.

In this example was used „-num 99“ and „-num 98“.

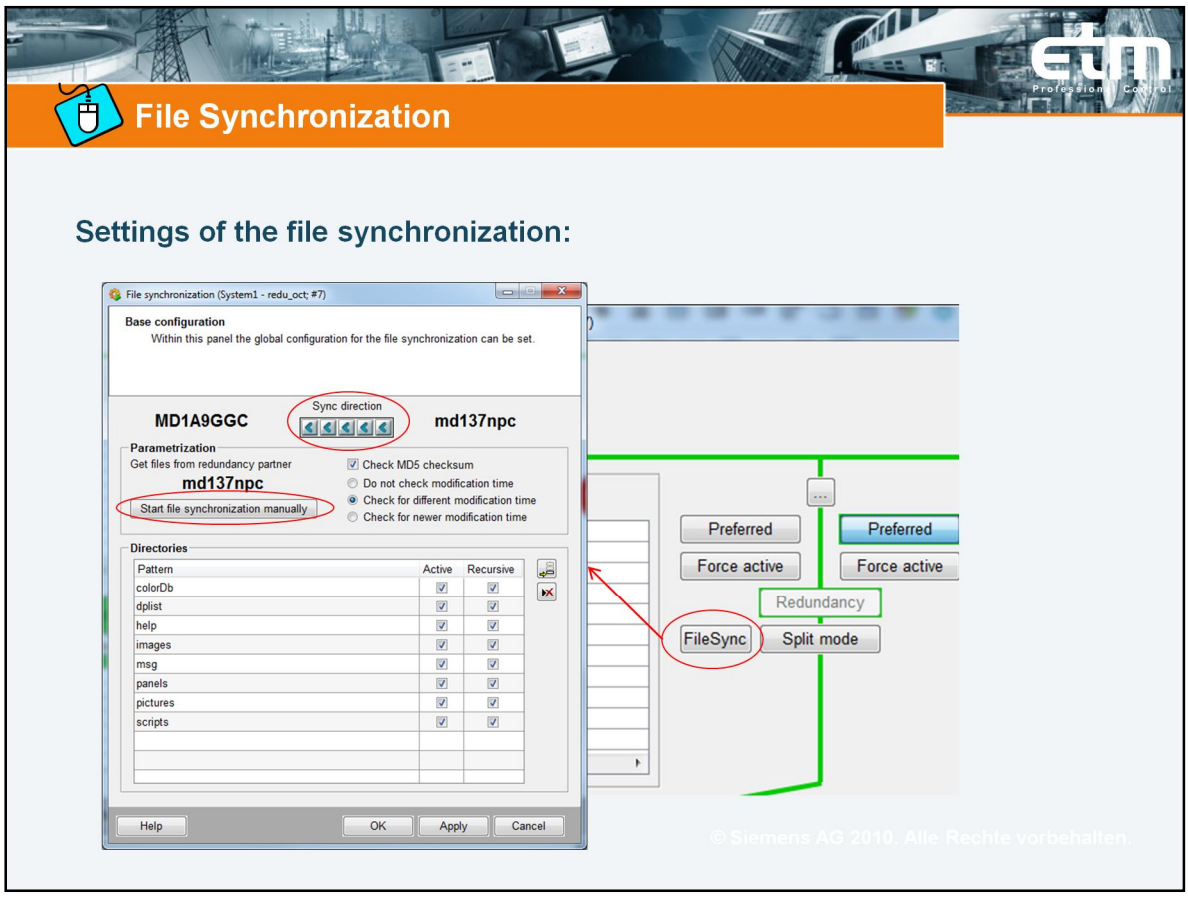
File Synchronization

View in the system overview and settings for file synchronization:



Note:

Now, when the system overview is opened, the **FileSync** button is present.



Note:

The settings and directories for the file synchronization can be configured in the file synchronization panel. By default the table contains all directories, which are recommended for the file synchronization.

It's possible to add more folder directories here. The "data" directory and the "backup" directory may not be synchronized, because these folders always contains the backup files.

!!! It's only possible to synchronize from the active red. Server to the passive one. !!!

Deleting and archiving files via the file synchronization

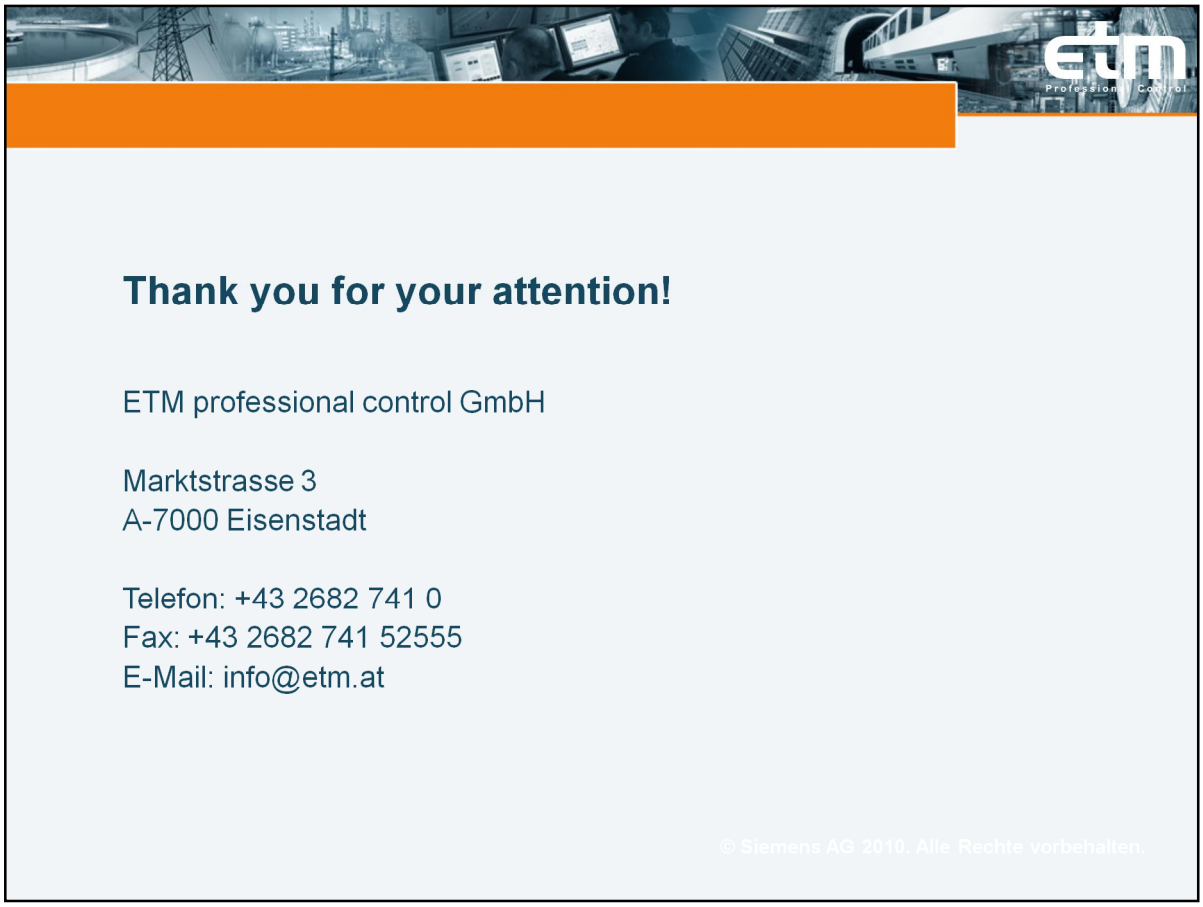
Files that are overwritten or deleted during the synchronization are stored in the data/backup project directory (if not another online backup path has been set for the backup on the host) with the name "projectFileBackUp_*<HostName>*_*<timeInSecondsSince1970>.zip*".

Security information

Siemens provides automation and drive products with industrial security functions that support the secure operation of plants or machines. They are an important component in a holistic industrial security concept. With this in mind, our products undergo continuous development. We therefore recommend that you keep yourself informed with respect to our product updates. Please find further information and newsletters on this subject at: http://support.automation.siemens.com.

To ensure the secure operation of a plant or machine it is also necessary to take suitable preventive action (e.g. cell protection concept) and to integrate the automation and drive components into a state-of-the-art holistic industrial security concept for the entire plant or machine. Any third-party products that may be in use must also be taken into account.

Please find further information at: http://www.siemens.com/industrialsecurity


Thank you for your attention!
ETM professional control GmbH
Marktstrasse 3
A-7000 Eisenstadt
Telefon: +43 2682 741 0
Fax: +43 2682 741 52555
E-Mail: info@etm.at
© Siemens AG 2010. Alle Rechte vorbehalten.