Requirements from Application section

Timing Review

Stephane Deghaye

29/02/2008

Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

What timing services

- Central timing events
  - Events distributed on the timing network
- Telegram
  - Set of info on what the machines are doing
- Run-time configuration
  - What is programmed (Sequence, BCD, SuperCycle)
  - What is played and why.
- Static configuration
  - Info on the different concepts (machine, user, telegram groups...)
- External conditions
  - Variables set externally conditioning the sequencing

Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

Usage summary (1/2)

- Central timing events
  - Connect to events to trigger computation.
  - Send events on request (LHC, Transaction).
  - ↳ For Logging, JAPC (Monitoring), SIS, Generic GUIs, Cesar...
- Telegram
  - Reception of one or several telegram groups
  - Human readable format
  - ↳ For Logging, SIS, FixDisplay, GUIs (Generic/Specific), LASER...

Usage summary (2/2)

- Run-time configuration
  - Retrieve SuperCylce length, active users.
  - Retrieve what was played and why (Normal/Spare).
  - For Fault Statistics (Logging), LSA, FixDisplay, Generic GUIs, Cesar...
- Static configuration
  - List of available users, central events, machine, telegram groups...
  - For LSA, Sequencer, Generic GUIs...
- External conditions
  - Acquisition and control
  - For Logging (read-only) and SIS (read/write)

Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

How to access the timing services Interface

- The timing services should be accessible through a JAPC interface
- Why?

  ➔ Homogenous view with the rest of the low-level control system
  - Usage of standard components
  - No need to learn a new API

How to access the timing services Non-functional requirements

The JAPC implementation must be:

- Reliable and easy to diagnose
  - Dedicated com & distribution should be avoided
- Extensible
  - Add another piece of info should be fast
- Able to use transparently a dedicated timing receiver
  - High reliability cases (SIS)

Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

Example

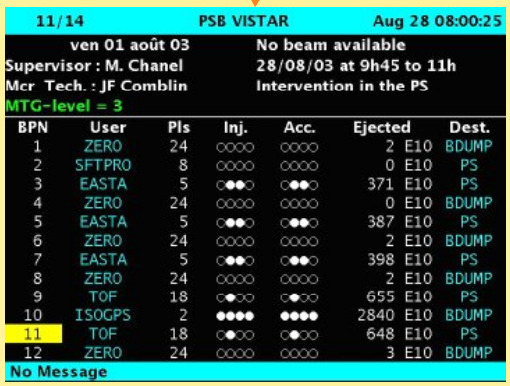
Possible use of the CPS user info (hypothetical structure)

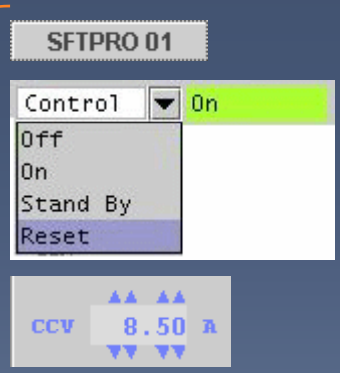
JAPC Parameter: "CPS-TELEGRAM/USER#CURRENT"

Specific code

Logging

JAPC Monitoring





Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

Implementation constraints

- Not strictly AP business but...
- Parameter structure needs to be defined in collaboration with AP.
- Need to understand how the components we use in our systems work.
  - Side effect
  - Reliability
  - Diagnostics
- → Strongly encouraged to use standard CO components such as FESA, CMW, SIS, Oracle...

Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

Future evolution – InCA

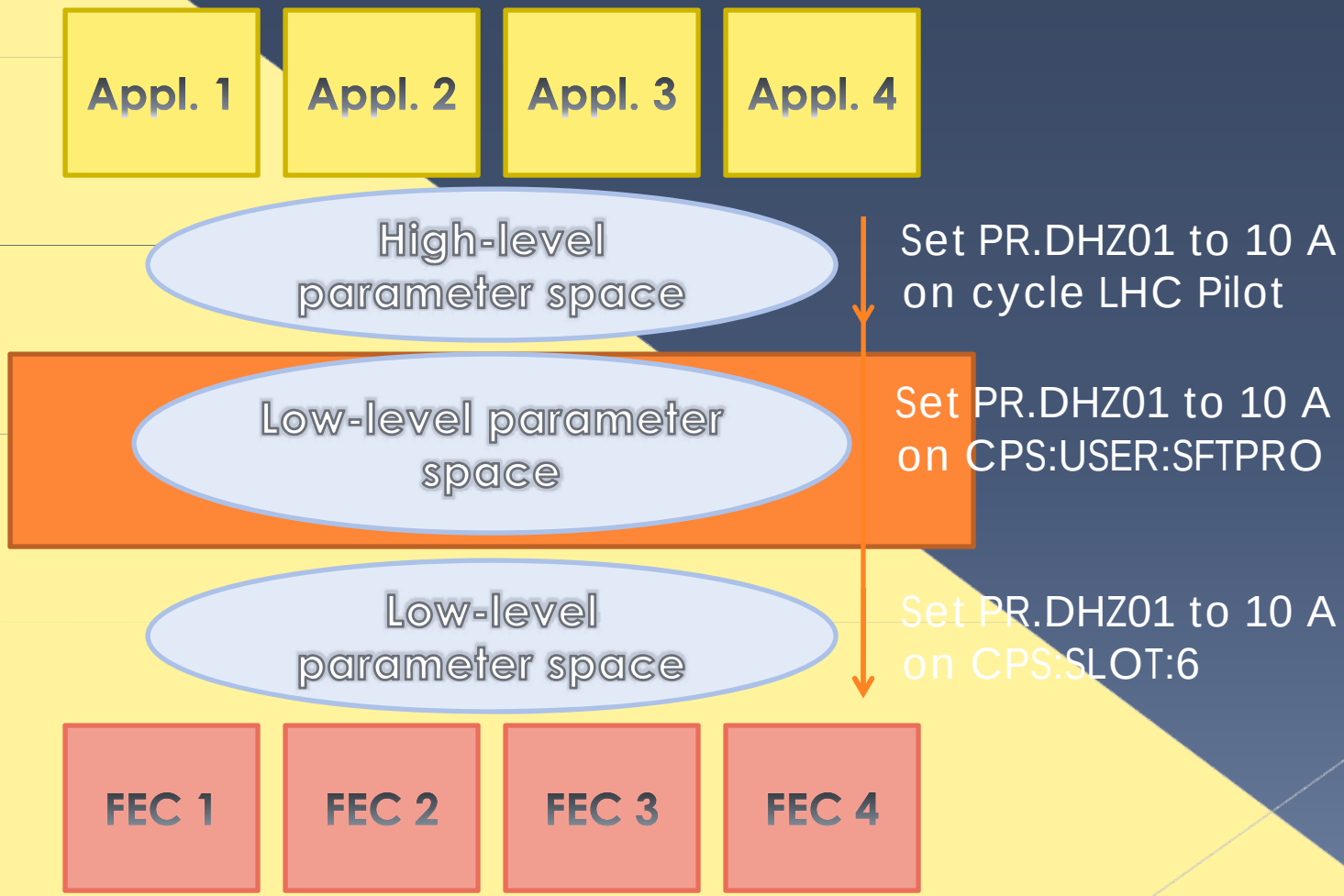
- Injection Control renovation going on
- InCA will bring new concepts & features
- Better integration of the high-level parts.

➔ Some constraints might be relaxed

E.g. The user name might be less visible

☝ Not for tomorrow

Future evolution – InCA



Agenda

- What timing services
- Usage summary
- How to access the timing services
  - Interface
  - Non-functional requirements
- Examples
- Implementation constraints
- Future evolution (InCA)
- Summary

Summary

- Set of timing services and their main usage identified
  - Central events
  - Telegram
  - Run-time & static configuration info.
  - External conditions
- JAPC interface to these services required
- Standard components must be used
- High-reliability installations must be possible

Question ?

Thank you for your attention

Example

Subscription to the CPS user using JAPC

```
public class TelegramListener implements ParameterValueListener{

    /**
     * Starts the subscription on the telegram group given
     */
    public void startMonitoring(String telegramGroupName){...}

    /*
     * Receives the updates and prints the result
     */
    public void valueReceived(String parameterId,
                              AcquiredParameterValue value){...}
}
```

Example - startMonitoring

```
/**
 * Starts the subscription on the telegram group given
 */
public void startMonitoring(String tlgGrpName){
  try {
    ParameterFactory fact = ParameterFactory.newInstance();
    Parameter parameter =  fact.newParameter(tlgGrpName);
    SubscriptionHandle handle = parameter.createSubscription(null, this);
    handle.startMonitoring();
  }
  catch (ParameterException e)  {
    System.out.println("Can't subscribe to " + tlgGrpName);
  }
}
```

Example - valueReceived

```
/*
 * Receives the updates and prints the result
 */
public void valueReceived(String parameterId,
                          AcquiredParameterValue value){
  String user =
    ((MapParameterValue)value.getValue()).getString("value"));

  System.out.println("Got " + user + " from " + parameterId);
}
```

Example - result

```
TelegramListener() listener = new TelegramListener();
listener.startSubscription("CPS-TELEGRAM/User");
```

-- Output example

Got SFTPRO from CPS-TELEGRAM/User

Got SFTPRO from CPS-TELEGRAM/User

Got AD from CPS-TELEGRAM/User