A selective overview of the FLASH control system for operators

Tim Wilksen
FLASH Control System Overview
Hamburg, 14.07.2017
Outline

> Essentials
> Tools
> Data Acquisition System
> Some Hints On Failure Detection
Essentials

- Concepts
- System Layout
- Server
Essentials – Basic Concepts

- Designed 20 years ago with focus on operating **accelerator facilities**
- The basic entity is a **device server** representing a (hardware) device
- Object-oriented design paradigm where a device is an instance of an object-oriented class
- Server holds several device instances as **locations**
- Device instance contains the **properties** of a (hardware) device
- Properties are data objects and access points on the communication network
- Device definition and configuration is done on server only and transparent to the communication network therefore enabling a **distributed architecture**
- Control system or device properties are called **DOOCS Server Properties**
- The naming scheme adheres to a four-string identifier
  
  **Facility/Device/Location/Property**
Essentials - Server

- Server core API implemented as C++ library
- Client-Server model-based
- Multi-threaded – update thread per location, interrupt-driven threads, …

Communication
- Network communication based on Remote Procedure Calls (RPC) with XDR data representation
- Additional server interfaces (hardware or protocols) as separate libraries
- Timing system information via device driver interface or ZeroMQ callback

Configuration
- **Local server configuration file** defines server devices (and locations)
- Configuration mechanism with save and restore functionality i.e. keeps device settings
- Locations and properties can be added and removed on-the-fly

Archiving
- Extended archiving functionality local to the server (a.k.a. histories or .HIST properties)
- **Long-term archiving** thru flexible filter algorithm
- **Recorder server** with long-time archives (lifecycle of accelerator facility)

Server access authorization and privileges
- Implemented via UNIX user id and group id on a per-server basis
- Fine-grained XML configuration on property level

Logging capabilities for debugging and access (location and property)
- Conversion of raw data to physical implemented as properties -> polynomial parameters (POLYPARA)
- Algorithms for calculating mean or RMS values -> CALC property
Essentials – Server I/O Support

Network

Device Server

Fieldbus, Hardware, …
Essentials – Server Communication Network Protocols

- **DOOCs** (RPC)
- **TINE**
- **EPICS**
- **ØMQ** (Push protocol only for server notification)

Network → Device Server → Fieldbus, Hardware, …
Physical Device Server

I/O Device Server

Device Node

Device Driver

AMC

Network

MicroTCA Backplane - Triggers and Clocks

© K. Rehlich
Tools

- C++ Client API
- Java Client API
- Python Bindings to C++ Client API
- MATLAB Bindings to C++ Client API
- JDDD
- RPC Utility
- Watchdog
- MicroTCA
- Network
Tools – Command Line Utilities

> Command Line Interface Tools (CLI Tools)

- Using DOOCS C++ client API
- doocsget and doocsput – lots of options for retrieving and formatting
- dhisttool for examining, maintaining and repairing archives

```
[mcswilksen:~] wilksen% doocsget -a -c "XFEL.UTIL/TUNNEL.TEMPERATURE/XTLL3/HUMIDITY.HIST" -H
# Address : XFEL.UTIL/TUNNEL.TEMPERATURE/XTLL3/HUMIDITY.HIST
# Data Type : A_TS_FLOAT
# TDS array length = 2000
2017-06-21 13:17:16,821 1498043836 821 0 : 3.9998779e+01
2017-06-21 13:20:58,382 1498044058 382 0 : 4.0047611e+01
2017-06-21 13:22:39,093 1498044159 93 0 : 3.9998779e+01
2017-06-21 13:22:49,164 1498044169 164 0 : 4.0047611e+01
2017-06-21 13:23:09,306 1498044189 306 0 : 3.9998779e+01
<...>
[mcswilksen:~] wilksen% doocsget -c "XFEL.VAC/ABSORBER/XTD2_V10000M/STATE"
OPENED
```
Tools – Python 3 Command Line Tools

- Python bindings are using the C++ DOOCS client API
- Python 3 only, specifically using the Anaconda distribution to simplify cross-platform and (!) cross-control system support
- Only client implementation, not yet all data types are implemented
- Basic read call in Python using the pydoocs package:

```python
>>> import pydoocs

>>> toroidData = pydoocs.read("XFEL.DIAG/TOROID/TORA.25.I1/CHARGE.SA1")

>>> toroidData
{'macropulse': 1412481288, 'channel': 'XFEL.DIAG/TOROID/TORA.25.I1/CHARGE.SA1',
'timestamp': 1498242907.235073, 'type': 'FLOAT', 'data': 0.5049999952316284}
```
Tools – MATLAB Command Line Interface

> Implemented with DOOCS C++ client API as MEX functions
  - doocsread
  - Doocswrite

[mcswilksen:~] wilksen% matlab2016 -nodesktop -nosplash

    < M A T L A B (R) >
    Copyright 1984-2016 The MathWorks, Inc.
    R2016b (9.1.0.441655) 64-bit (maci64)
    September 7, 2016

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> result = doocsread('XFEL.DIAG/TOROID/TORA.25.I1/CHARGE.SA1');
>> result

result =

struct with fields:

    channel: 'XFEL.DIAG/TOROID/TORA.25.I1/CHARGE.SA1'
    data: 0.500050008296967
    timestamp: 1498461821.6088
    systemstamp: 1414670455
    type: 'FLOAT'
    error: ''
Accelerator FLASH MainTaskbar: Starting point for all accelerator operations
The Java DOOCS Data Display (JDDD) is an editor for graphical user interfaces to the accelerator control system (often called “panels”)

Create complex control system panels without programming

Main concept:
- Control system experts provide tool (JDDD) and generic panels (GUIs)
- System experts provide panels for their systems
- Operators can provide or enhance panels specific to operations
- Users provide panels for beamlines and beamline experiments (applies only for FLASH currently)

Rich set of widgets for representing control system objects
- Dynamic layouts and components with animated graphics
- Conditional widgets, dials, audio, colored indicator, location chooser,…
- Plotting widgets with integrated (simple) data analyses (mean, RMS, …)

Centrally hosted panels as XML files (Subversion repository with versioning)

Extended data manipulation with JavaScript (Mathematical operations)

Smart address generation with DOOCS Syntax, regular expression filter and global or local parameters

Huge number of options for plotting – domain, history, logic plots ...

HTML-based version of JDDD available for remote access via browser (https://jddd.desy.de)
Tools – JDDD Editor

Inspector
Available widgets/components
Component properties
Panel Editor

http://jddd.desy.de -> Help
Tools – Browser

- RPC (Remote Procedure Call) Java Tool to browse all DOOCS channels in DESY name space
- Use as an “explorer” tool to learn about the system
- Based on JDDD (it is actually kind of JDDD panel)
Tools – Browser (cont’d)

- Filter with regular expressions
- Plot all locations
- Read values as text (file)
- Copy selected channel address
- Read and set values

FLASH MainTaskbar -> Status -> Control System -> RPC Utility
Tools – Watchdog

> Watchdog on every node
- Manages and monitors all server processes
- Auto-restart of server processes
- Error reports
- Statistics reports

FLASH MainTaskbar -> Status -> Control System -> DOOCS Watchdog
Tools – Watchdog Overview

> Watchdog Overview i.e. FS-FLASH-D
Tools – Management MicroTCA Systems

FLASH MainTaskbar -> Status -> Control System -> MicroTCA Systems

Watchdog Status
Green == OK
Yellow == Error

MicroTCA Crate Status
Tools – Management MicroTCA Systems Crate Overview

MicroTCA System Crate Overview:
Status of each AMC and RTM
-> Reboot, restart
Tools – FLASH Network Monitoring

FLASH MainTaskbar -> Status -> Control System -> DOOCS System Status -> Network Status

- Red Switch -> Switch Down, call NOC and MCS4
- Flat I/O Rate: TINE Recorder Down -> Notify MCS4
Data Acquisition System

- General Layout
- Middle Layer Services
- FLASH DAQ Instances
- DAQ Tools
Middle Layer Servers or Application Servers

- **Middle layer server is a client** with respect to the buffer manager. Hence it has access to all subscribed event types and shot-synchronized data.
- Can not only read data (**consumer**), but also process it and write results back to the same event record (**producer**).
- Notified when requested data (subscribed channels) are available in shared memory.
- Can perform many kinds of computational tasks.
- Uses **MATLAB** or **C/C++ for calculations**, standard mathematical libraries are vastly available.
- Sufficient response time for many operational and automation purposes. Examples: Energy measurement, photon wavelength measurement, gas monitor detector server, LLRF server, orbit server.
- Two API options for ML server development.
Data Acquisition System – FLASH DAQ Instances

Seven DAQ instances at FLASH: FLASH MAIN DAQ, User DAQs, TEST DAQ (SFLASH)

Instance: MAIN DAQ

Traffic Lights == Health Status:
- Grey = Off/Idle
- Green = OK
- Yellow = Still OK
- Red = Take Action, check Status Panel

Disk Space: green or yellow = OK

Output Rates: OK if > 0

Input Rates: OK if > 0, If dip occurs -> FED problem

FLASH MainTaskbar -> Status -> Control System -> DAQ Status
Data Acquisition System – FLASH MAIN DAQ Status Panel

- Event Builder and Writer
- DAQ Middle Layer Servers
- Buffer Manager
- Collector
- Front-End Server

FLASH MainTaskbar -> Status -> Control System -> DAQ Status -> MAIN -> Status

Disk Status:
- Event Number should increment
- Lost Event Number should NOT increment

Loss Rates:
- Disk Status
- Event Number
- Lost Event Number

FED Sender Status:
- Green == OK
- Orange/Yellow == ERROR
- Light Grey == OFF
- Dark Grey == System/Process Down

Event Builder Writer Status:
- Photonius Server
- GMD Provider Server
- Photonanergy Server
- Photonwave server
- HYCA Server
- Pass Server
- Beam Profile
- HCM Server
- SLM HC Server
- SADC Statistics Server

Event Number
- Should increment
- Should NOT increment

ボタンの再起動
Data Acquisition System – DAQ Tools (Selection)

Data Retrieval (Non-Middle Layer Services)

- Several API to read and extract data: C++- and Java-based ones. On top of these: MATLAB (Octave, Mathematica), JDDD, DAQ Data GUI, Python-based one soon
- DAQdataGUI
- JDDD
- MATLAB

**DAQ channel FLASH.RF/LLRF.CONTROLLER.DAQ/8PORT.GUN**

![Graph of DAQ channel data]

**Java-based DAQ GUI**

![Graph of DAQ channel data]
Failure Detection – “Red JDDD Object”

“Red” JDDD Object?
(!= Watchdog otherwise go to “System Down”)

Yes

Shows “Unavailable Server”?

Yes

Check Watchdog Panel

No

Shows “Java Script Error”?

Yes

Server Down?

Yes

Try To Restart Server via Watchdog Panel

No

Yes

Notify or Call System Experts (or last resort, MCS4)

Server Running Again?

No

Yes

Restart JDDD Session

What?

Yes

Check FEC Status

No

Server Down?

Yes

No
Failure Detection – “System Down”

System Down? (== Watchdog Red)

Yes

Is it a MicroTCA System?

Yes

Is the Crate Status Panel Accessible?

Yes

Is it possible to log on with ssh?

Yes

Try To Restart Watchdog via command line

System And Watchdog Running Again?

No

Power-cycle CPU AMC Via Crate Status Panel

System And Watchdog Running Again?

Yes

Power-cycle MicroTCA System Via MCH CLI

System And Watchdog Running Again?

No

Notify or Call System Experts (or last resort, MCS4)

No

OK
Failure Detection – “DAQ Traffic Light Red”

1. Traffic Light For DAQ Instance “Red”?
   - Yes: Disk Status on Instance Overview Red?
     - Yes: Check DAQ Status Panel
     - No: DAQ Server Not Running? (= ML Server)
       - Yes: Restart ML Server After Cycling It To INIT State Via Watchdog Panel
       - No: DAQ Server Not Running? (!= ML Server)
         - Yes: Restart DAQ Instance Via DAQ Status Panel
         - No: Notify or Call System Experts (or last resort, MCS4)
   - No: Check DAQ Status Panel

2. Disk Status on Instance Overview Red?
   - Yes: No
   - No: Yes

3. DAQ Server Not Running? (= ML Server)
   - Yes: OK
   - No: No

4. DAQ Server Not Running? (!= ML Server)
   - Yes: Yes
   - No: No
Failure Detection – “DAQ Traffic Light Red” JDDD Version

Flow chart for failure detection and resolution implemented as JDDD panel with active buttons.
References

Web reference:

• DOOCS Web Page: http://doocs.desy.de

• FAQ: DOOCS Web Page -> FAQ or https://ttfinfo.desy.de/FLASHWiki/Wiki.jsp?page=FAQ

Request Tracker (Email) also via Logbook:

• doocs-rt@desy.de