IDENTIFYING CONTROL EQUIPMENT

M. Clausen, M. Möller DESY, Hamburg, Germany

Abstract

The cryogenic installations at DESY are widely spread over the DESY campus. Many new components have been and will be installed for the new European XFEL. Commissioning and testing takes a lot of time. Local tag labels help identify the components but it is error prone to type in the names. Local bar-codes and/or Data Matrix codes can be used in conjunction with intelligent devices like smart (i)Phones to retrieve data directly from the control system. The developed application will also show information from the asset database. This will provide the asset properties of the individual hardware device including the remaining warranty. Last not least cables are equipped with a bar-code which helps to identify start and endpoint of the cable and the related physical signal. This paper will describe our experience with the mobile applications and the related background databases which are operational already for several years.

INTRODUCTION

For those familiar with day to day operations of bigger cryogenic plants it is a usual problem to work in the field and identify mal functional equipment. Several questions arise immediately: Do we have a spare? If yes – where can I find it? If there's no spare or if you want to fill up the stock you need to find out when specific equipment was purchased – and where (!).

So you have to find out what kind of equipment it is and which specific version of a specific kind. Then you will try to find the folder where you have gathered your purchase orders. But in which year did you buy this equipment? – And from which vendor?

All of these questions come easy when you have tagged your equipment upon arrival – before it has been installed in the field. This was the starting point of our internal asset database. We collected the data of our new equipment in a database and each component was tagged with a unique number. In the beginning it was a label in clear (ASCII) text form. We could read the ID and check for the information in our database from a terminal. For a short time we used barcode. This caused trouble when using barcodes on small devices with small barcode printouts. Nowadays we are using Data Matrix code which can be easily and clearly identified by most apps on nearly any smartphone.

Choosing the right (Tag-) Code

As mentioned above we have chosen the Data Matrix code for our tags. The advantage is clearly that it is still readable even when printed out on a small tag. There's often the question whether QR-Code wouldn't be a better choice? (See Fig. 1 for a comparison of the two different codes). QR can be used to store more complex data like a complete http address or more properties of the equipment. This would allow sending the QR-Code directly to an http-server and getting the results back. We did not see an advantage in this approach. Http servers tend to change. Application names tend to change. Even technologies accessing the information will change over the years – at least for time periods of our projects which typically run for more than one decade.

Figure 1: Comparison of Data Matrix and QR-Code displaying the same string: "1-203074".

In our case we want to keep it smart and easy. Just the ID of the equipment shall be kept in the tag. All of the other information shall come from the database storing all the metadata and keeping the relations to other data sources.

ACCESSING DATA

Different locations require different access methods. The operators in the control room refer to the equipment from the control systems point of view (namely the channel– or device names they are working with). The technician in the field on the other hand looks at the 'real' equipment and wants to know which channel is connected to this equipment and which read back value is the actual one.

Control System Studio (CSS)

The operator consoles in the cryogenic control room are running Control System Studio (CSS). CSS is an extensible set of Eclipse plugins written in Java.

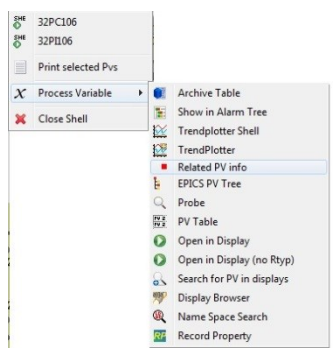

Figure 2: Contribution menu in CSS.

ISBN 978-3-95450-139-7

'Information on your fingertips' is the term created for easy access to additional information related to the current process channel the operator's mouse is pointing to. The technical realization of this functionality is a contribution menu (Fig. 2).

Each CSS plugin which can handle channel information is registered in the contribution menu. In practise the operator is working with a channel or device and wants to get more detailed information. This could be a request to display archived data, to see the alarms of the last 24h or – to find out what the related equipment looks like, where it is installed and whether spare parts are available.

Here come our equipment databases into play.

DATABASES

Equipment databases have a long history in control system environments. Some of these only contain hardware information. Other databases also contain information about the devices with respect to the machine lattice of an accelerator or similar information.

In Many cases these kind of information are stored in the same database structure tightly connected to each other with strong relations between the individual tables. Whenever changes apply to any table in this database driven environment most – if not all – other components are affected. We went through the same learning curve and – as a result – we decided to keep individual information in individual database tables which are loosely coupled to each other. Coupling is performed by IDs in the form of the same strings which are defined in the individual tables – or set of tables. This implies that not every table is related to each other table but in many cases by another relation in another table.

Generic Database Access

Information separation is not the only important design criteria. Another criterion is the access to the data. In the end a client application just wants to get search results but it should not have any specific knowledge or code to select the information from all of the data sources.

This resulted in a generic service architecture which hides all of the complexity from the client application. When a client application sends a request for a control system channel, a barcode-number or any other specific ID, a dispatcher will take care of it. The dispatcher will 'ask' all of the registered services whether the content of the string was known to that specific service. The service itself will try to find a match with any selection ID stored in it's tables. Whenever a match is found it will prepare the answer for the dispatcher. If a second ID is stored in conjunction with the initial ID it will try to gather also that information through the dispatcher from yet another service. In the end the client application will receive a complete answer with all possible related entries in all registered services.

Having this generic service layer in place it is easily extensible with new data sources. The only criteria such a service must fulfil is that it must match with any other ID of any other service already registered.

The list of the actual implemented equipment databases and the main access IDs are the following:

Asset Database

The asset database is used to identify all equipment in the cryogenic control system starting from individual I/O cards to mainframes or RAID systems. In every case it is used to store purchasing information, the state of the equipment (in stock, in repair, installed (in) …). The information is installed in is important to be able to actually find this equipment at DESY.

In practise the information has helped to find out whether spare parts are still available or where the component was purchased and when.

The search IDs are: Asset-#; (DESY)Inventory-#

DESY Inventory

The DESY inventory number has references to the SAP database. Purchase IDs and related information is available here.

The search IDs are: (DESY)Inventory-#

Searching across Data Sources

Having the two data sources in place it is now possible to get internal information from the asset database when the inventory number is known and vice versa to find out the inventory number from the asset ID entry.

EPICS Channel and I/O References

EPICS IOC configurations (called EPICS databases) are created at DESY by means of a special configuration tool. The database configuration tool (DCT) is designed to create hierarchical structures of EPICS records. These structures allow a device oriented approach where prototypes of record based processing chains can be nested into each other and as a result creates a flat EPICS database configuration file. The DCT configuration information on the other hand is stored in an ASCII file which is under cvs version control.

The reference from the EPICS records to the I/O hardware is performed through so called IO_NAMES. These are unique names in the I/O database. When the EPICS database is created, the IO_NAMES get resolved and the underlying addresses of the I/O equipment replace the IO_NAMES in the individual instances of the EPICS records. This way the database engineer and the equipment engineer can work independently from each other. The 'clue' between the two are the IO_NAMES.

The search IDs are: EPICS Record Name; IO_NAME

Device Database

The configuration of Profibus DP/ PA I/O devices is a complicated task. The commercial configuration tools are basically only available for Windows based workstations. In this case the created configuration file would be in binary format and specifically designed for a specific Profibus hardware board. In our case we have to

ISBN 978-3-95450-139-7

configure the memory layout of all the Profibus devices for a VxWorks based control system in a form that the driver of the board can read it and on the other hand the addresses in the dual ported memory can be configured in the EPICS database. The configuration tool fulfils these requirements. It creates a configuration file for the driver and provides memory address layout information through a special service layer. The most complicated task is parsing the configuration of all the so called GSD files and keeping this device specific configuration 'in mind' when configuring the Profibus nodes. The link between the EPICS database and the device database are the IO_NAMES. These are unique in the whole system, - like the EPICS record names. A service resolves the IO_NAMES in the device database and returns the Profibus memory address in the dual ported memory.

The device database also keeps documentation for the specific I/O devices and stores also the current Asset-# of the real device as it is currently installed in the field.

The search IDs are: IO_NMAME; Asset-#

Cable Connections

In former times we had several 'special events' where the tram in the HERA tunnel was scratching along the tunnel by accident. Unfortunately there were some cables mounted on the wall and the cables were cut. Of course it was essential to know which components were affected. This information would help to decide whether emergency operations could continue. In the cryogenic controls case: Can we keep the magnets cold? At that time we were not good in documenting cable numbers, attaching labels on the cables and providing an electronic database of the cables and the equipment connected to them.

This situation shall be avoided in the future. A cable database which actually keeps the information about the devices connected to the cables shall help for future maintenance and hopefully no incidents. But not only incidents are important to handle. Also the question in case of maintenance work is important and last not least if you are working with a device and want to know to which cable this specific device is connected.

The search IDs are: Cable-ID; IO_NAME

EPICS Data

A very special type of database access is the EPICS live data service. This service obviously provides live data from the control system. Another important information is the actual state of the observed channel (connected, disconnected, invalid …).

The search IDs are: Channel name

Database Extensions

The generic approach based on OSGI services allows an easy extendibility of the data sources behind the dispatcher. A new data source only has to share one search ID with any other service and it can be integrated into the system.

USER INTERFACES

There are currently two user interfaces operational. One app for the iPhone and another integrated into CSS.

iPhone App

The development of a generic service layer for the data sources was driving the idea to develop a generic user interface on the iPhone. 'One app serves all' is the idea behind this generic interface. Only a handful of components are supported in the display. These can be dynamically combined. The display is configured by an XML data stream which gets locally parsed. The whole workflow looks like this:

- Scan a barcode or Data Matrix code. – Or just type in the code, name or whatever is available.
- Send request to a web-service which is using the already mentioned dispatcher to query all the defined OSGI Services.
- Services and dispatcher prepare an XML data stream back to the app.
- The app is parsing the XML stream and displays the content with the predefined widgets on the iPhone display.

Figure 3: Generic iPhone app with it's basic components.

Figure 3 shows the generic iPhone app. The basic components are:

1. Display of the currently observed ID
2. List of tag/ value pairs

ISBN 978-3-95450-139-7

3. List of tag/ value pairs with a link to another page. The additional information will retrieved by another request to the web.-service
4. (Not shown here) Text entry field(s)
5. A text field with describing text information

The type '3' links can be http links or links to another internal app in the iPhone. In the case of an EPICS channel this could be a link to the ArchiveViewer (Fig. 4).

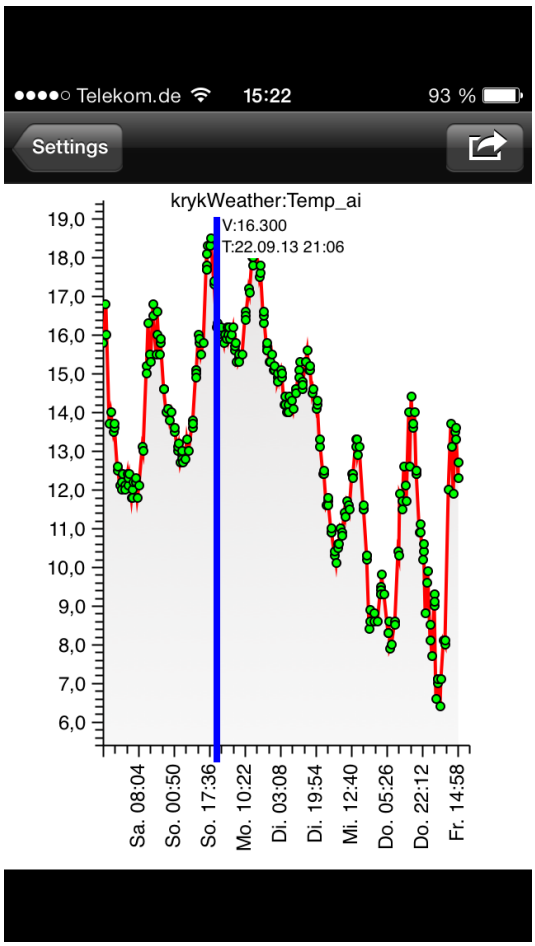

Figure 4: Archive Viewer App on the iPhone.

CSS User Interface

The development on the CSS side is still 'work in progress'. Some basic features already exist like the entry in the contribution menu (Fig. 2) and the resulting pop-up window which displays information from three different data sources (Fig. 5):

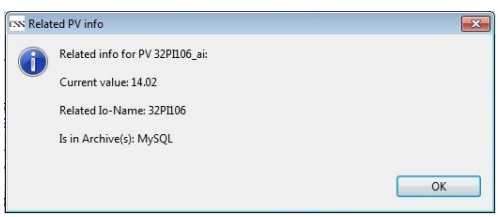

Figure 5: Pop-up window called from contribution menu.

The second example shows the list of documents which are available for a specific device type. The service identifies a valid channel in the I/O configuration -> the device type -> and selects all documents available for this device type (Fig. 6).

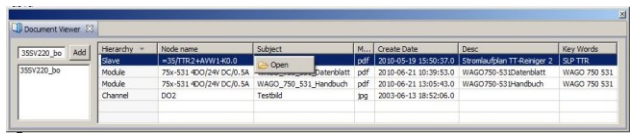

Figure 6: Documents available for a specific device type. Selected by channel name.

CONCLUSIONS

The complexity of a single database with several data sources in various tables which have tight relations to each other has been avoided. The structure of each data source has been kept as simple as possible. Each new data source just needs one ID in common with any other data source to integrate it. A service layer has been developed which covers data source specifics.

A generic iPhone app has been developed which handles any answer from the service dispatcher by parsing an XML file. No specific code for a new data source is necessary.

The integration in CSS is achieved by CSS contribution menus. Data source specific applications can be called from CSS to display the full dataset specific properties.

ISBN 978-3-95450-139-7