Channel Configuration Management

A. Lüdeke, PSI, Switzerland

Abstract

The reference of actor / sensor values in control systems is done by names, so called channels. High level application have these channel names either hard-coded or include them by configuration files. An automatic offline availability check of all channels used in any configuration file of a high level applications is desired for the SLS. The check has three branches: Each channel server registers all served channel names in a central database. Each configuration file is uploaded to the same database and can be downloaded to the control system from the database.
In addition to the check of configuration files many useful information can be retrieved from that system. The paper describes the status of the implementation and the performance of the system.

1 INTRODUCTION

The SLS is a 2.4 GeV third generation synchrotron light source. Like for other light sources, the availiable time for testing new applications on the real machine is very short. In the EPICS control system, each IOC provides a number of channels which can be used by any number of applications. One recurrent problem for the control system is, that changes in channel names are not communicated between developers or that inproper loaded databases do not provide the full set of required channels. Since these problems are detected at startup of the application which requires the channel, this type of problem often lead to a significant delay of the machine startup and therefore to a shortening of the time for machine studies.
To overcome these problems we generate database tables of the required and the availiable channels in order to allow automated offline checks to detect missing channels. As an additional aspect, we can check all used and loaded channel names with respect to the naming convention and the hierarchical defined device-property pairs used to access EPICS channels via CDEV.
The following sections will describe the used mechanisms:

- Automatized uploading of EPICS channel databases to an oracle database,
- uploading channels from configuration files to an oracle database and
- checks and information retrieval using the above data.

2 UPLOAD CHANNELS FROM EPICS

Each EPICS IOC runs a startup script to load the EPICS databases. At the SLS this startup script has a generic part, identical for each IOC. We've added a command after the initializion of the EPICS databases, to dump all existing channels to files, sorted by the record type. This is done using the standard EPICS command "dbl" for each record type. The IOC then starts a script on it's boot PC, using the vxWorks command "rcmd", to upload those files to the oracle database. The script `db2odb` splits the channel name into device and property, according to the SLS naming convention at the colon ":", and uploads device, property, record_type, ioc, load_time and load_date to the table IOC_channels. If the name goes away or the record type changes, the two fields in the database deleted_date and deleted_time of the existing row will be set to the upload time and a new line for that channel will be inserted. Figure 1 shows a flow chart of the script. The table IOC_channels contains only information about the existing channels and their record types. The existing fields of the different records and the type of these fields are

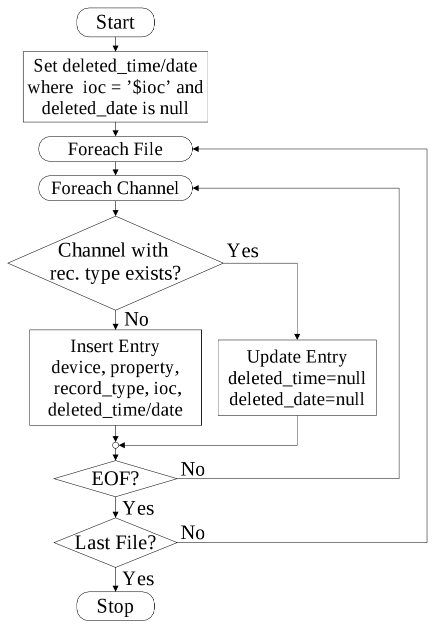
Start
Set deleted_time/date where ioc = '$ioc' and deleted_date is null
Foreach File
Foreach Channel
Channel with rec. type exists?
Yes
No
Insert Entry device, property, record_type, ioc, deleted_time/date
Update Entry deleted_time=null deleted_date=null
EOF?
No
Yes
Last File?
No
Yes
Stop

Figure 1: Flow chart of the script `db2odb` to upload EPICS channels to oracle.

stored in the table Record_Type_Definition. A second script `dbd2odb.pl` is used to upload this information from the system wide database definition file to oracle. A join of the tables IOC_channels and Record_Type_Definition allows to retrieve all accessible fields any channel.

3 UPLOAD APPLICATION CONFIGS

Application configuration files are used for many generic applications in the EPICS enviroment. These files contain a wide variety of information and are often maintained with graphical tools, like the adl files for medm.

For our purpose we just wanted to extract the information about the required channels into the relational database. The file itself is stored as a BLOB[1].

Each type of configuration file can require a different type of parsing in order to retrieve the required channels. For some types, like medm, the file itself can be a template and does not include all information to construct the channel names. In those cases additional command line arguments are required to determine the actual channel names. The tables Application_Config_Files and Application_Config_Channels are used to store the information. The first one is used to keep the filename, the file itself, a cvs version number and the date and time of the upload. The second table keeps for a filename / macro variable combination all used channels in form of device-property-field tupel. The script `xpv` loads a configuration file to the table Application_Config_Files and stores all channels used by the file in the table Application_Config_Channels.

The general approach is, to start the parsing with the command lines used for operation. At the SLS one tool is used for the startup of all applications: the programm launcher.tcl. All applications that can be launched by that tool have a startup command line defined in the launcher configuration files. The script `lcf2odb` parses these files and call `xpv` for each command line.

Currently the script `xpv` can parse configuration files for the following applications:

- **medm/dm2k:** GUI builder tools, supporting macro substitutions and recursive calls of other applications and related panels.[1][2]
- **StripTool:** Tool to display channels versus time, either from realtime or from archived data.[3]
- **Channel Archiver:** Tool to archive channels.[4]
- **Alarmhandler:** Shows a channel alarms hierarchy.[5]
- **save/restore:** Tools to save and restore machine states to files. Request files define list of channel to save to a snapshot file.[6]
- **psmController.tcl:** RF expert control panel. Provided together with the turn-key 500 MHz RF stations by the company Thales (formerly Thomcast) for the control of the five RF plants.[7]

[1]BLOB: Binary Large Objects

- **panel.tcl, bar.tk, xyplot.tk, strip.tcl, motor.tcl:** Generic user interfaces to display or control groups of channels, build in-house.

The same configuration file can be used with different types of macro variable settings. The file is only stored once in Application_Config_Files since it does not change. The required channels are stored in Application_Config_Channels together with the given macro substitution variables, since the same file can access different sets of channels depending on the substitution variables.

4 USAGE OF UPLOADED DATA

The information can be used for various kinds of checks and retrievals. We will illustrate a few of the applications in the following paragraphs.

4.1 Channel Availability Check

The original purpose was to have an automatic check, if all required channels are available for each used configuration file. The check is simply a single select statement:

```
select filename,macrovar,
       device,property,field
from Application_Config_Channels a
where not exists (
  select *
  from IOC_Channels i
  where a.device=i.device
    and a.property=i.property
    and a.field in11 (
      select field
      from Record_Type_Definition r
      where i.record_type=r.record_type
    )
 );
```

which gives a list of device-property-field tupel in all files that are not supported by any IOC.

4.2 Channel Availability History

The database does allow you to search for channels that are no longer supported by any IOC. This may help to find out which change did result in the disappearance of a missing channel. The Time of deletion and the name of the IOC that formerly supported the channel are kept.

4.3 Check of CDEV classes

The SLS naming convention of the SLS constructs a channel name from a devices name and a property, concatenated to a channel name using a colon character. The devices are according to a strict naming convention and all existing devices are defined in the so called master table. We assign each device to an interface class. This class defines all public properties of any device of that class. The classes can inherit properties from each other in a hierachie. The master table, the interface class hierachie and the relation is maintained in the database.
The definition of the valid properties for a device were not

coupled to the EPICS databases. A system responsible had to add new channels to both and also to the configuration files, whenever required. The new scheme allows now to check the validity of existing device names and to update the interface classes according to the required channels for the existing applications.

4.4 Other Checks

Some checks are already required at the upload. For example the reference of non existing configuration files is directly reported. This is an important feature for related medm displays.
Some useful runtime information can be retrieved from the database, too. For example if some channels are not available at startup of an application, a simple query can determine the IOC which should have been provided those channels. It can also tell if those channel were still provided after the last reboot of that IOC, or if they went away afterwards, for example due to a crash of the IOC. And it can provide a complete list of all channels that were discontinued at the last reboot, which in turn can help to derive the origin of the problem.
The low level application developer has also know the possibility to find out, who is using his channels. This is often needed in case he has to change the channel names in his EPICS databases.

5 PERFORMANCE

The Channels upload script `db2odb` was tested on EPICS databases running on two simulation IOCs, with about 16 thousand records each. The delay in the boot time of the IOCs is small, in the order of 20 second[2] of a total boot time of two minutes. This delay is *after* the IOC is already fully operational. The upload to the oracle database is done asynchronously from the boot PC and takes about 90 seconds.
The scripts `lcf2odb` and `xpv` were tested on the configuration files for the real accelerator. The parsing and upload of 590 configuration files which accesses more than 20 thousand different channels is done within 4 minutes.

6 SUMMARY AND OUTLOOK

All described scripts and features were implemented with the rather limited manpower of about two man weeks. In the next steps the automatic upload will be implemented for the real machine. By end of the year all IOCs should send their channels to oracle and all highlevel applications in production will be automatically checked against the existing channels.
The channel configuration database is steadily evolving new features which will increase the reliability of the application development. It's features are very helpful to test the correctness of new applications, to provide an overview for the control system and even for the analysis of runtime failures of the accelerator control system.

[2]IOCs are 300 MHz PowerPC VME boards, PCs were 350 MHz Pentium3

7 REFERENCES

[1] K. Evans, "*MEDM: Motif Editor and Display Manager*", http://www.aps.anl.gov/epics/extensions/medm/index.php

[2] T. Birke, "*dm2k Home Page*", http://www-csr.bessy.de/control/SoftDist/dm2k/

[3] C. Larrieu, "*StripTool Home Page*", http://www.jlab.org/ larrieu/work/StripTool/info.html

[4] K. U, Kasimir, "*Channel Archiver Home Page*", http://lansce.lanl.gov/lansce8/Epics/Archiver/Default.htm

[5] J. Anderson, "*Alarm Handler*", http://www.aps.anl.gov/epics/extensions/alh/index.php

[6] J. Winans, "*caSaveRestore Users Guide*", http://www.aps.anl.gov/epics/extensions/casr/casr.960325.html

[7] W. Portmann, "*The Thomcast RF user interface*", http://www.sls.psi.ch/controls/help/operators-manual/rf/expert.html

A TABLES

Table 1: IOC_Channels

Column	*Type*
Device	varchar2(23)
Property	varchar2(16)
Record_Type	varchar2(16)
IOC	varchar2(23)
Load_Time	varchar2(8)
Load_Date	Date(9)
Deleted_Time	varchar2(8)
Deleted_Date	Date(9)

Table 2: Record_Type_Definition

Column	*Type*
Record_Type	varchar2(32)
Field	varchar2(5)
Field_Type	varchar2(12)

Table 3: Application_Config_Files

Column	*Type*
Filename	varchar2(64)
Config_File	blob
Upload_Time	varchar2(8)
Upload_Date	Date(9)

Table 4: Application_Config_Channels

Column	*Type*
Filename	varchar2(64)
Macrovar	varchar2(256)
Device	varchar2(23)
Property	varchar2(16)
Field	varchar2(5)