IFC TO FESA GATEWAY: SMOOTH TRANSITION FROM GSI TO FAIR CONTROL SYSTEM

G. Jansa, I. Kriznar, G. Pajor, I. Verstovsek Cosylab, Ljubljana, Slovenia
R. Bär, L. Hechler, U. Krause, GSI, Darmstadt, Germany

Abstract

Present GSI control system uses an in-house developed CORBA based middleware called IFC. For FAIR project that will be built on the GSI site, a new control system is foreseen. New devices that are being integrated into the control system will be developed in CERN's Front End Software Architecture (FESA). In this article, an IFC to FESA gateway will be presented. The gateway provides an intermediate layer that is able to talk to FESA device servers on one side and provide their functionality to existing IFC clients. The gateway will allow coexistence of FESA front-end implementations and existing GSI device servers and clients, providing a smooth transition path to the future FAIR front-end environment. New GSI and FAIR devices that will be implemented in FESA will have to match GSI standards for nomenclature and device modelling. Exact match of new devices is not possible due to different hardware and software architecture of the new system, therefore a gateway solution is required. The gateway can translate the complete device model, including conversion from FESA to IFC data types.

INTRODUCTION

The Facility for Antiproton and Ion Research (FAIR) will be built on the GSI site. The present GSI UNILAC and SIS18 accelerator ring will be used as a injector for new accelerator installation. FAIR facility will be operated in a multiplexed mode, meaning that multiple experiments will be supplied with several different ion types simultaneously in a similar way as in the old GSI accelerator. The existing control system at GSI is well adapted for the present needs, but due to technological requirements a new control system will be used for FAIR. For front end part of the control system CERN's FESA was chosen [1].

For the transition period, when new devices are implemented in FESA but old GSI installation and many clients in control room will still use IFC, an intermediate layer of software will be required which will act as a glue between new FESA device servers and old IFC clients and device servers as shown on figure 1.

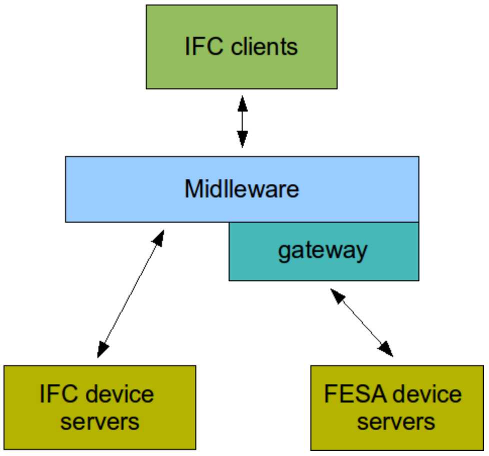

Figure 1: Coexistence of IFC clients and device servers and FESA device servers.

IFC AND FESA

IFC Description

IFC is an in-house developed CORBA based control system which has a narrow interface between device servers running on front-ends and clients running on remote machines accessing properties on device servers. By narrow interface we mean, that there is only one method of a given type (synchronous, asynchronous and connected read/write/call) which can control all of the device server's properties by specifying the property name as an argument of the method. IFC supports synchronous and asynchronous read, write and call (call is an action and can be seen as write operation without data) as well as connected read, write and call which are used for monitoring the property or for periodic setting of specified property.

Data written to or read from property is transported using data container which is a vector of values of supported data types. The values in container are not named and therefore the information of the meaning of specific value is given separately as XML description.

FESA Description

FESA uses CMW (Controls Middleware) which is CORBA based communication middle layer between FESA device servers and higher level applications. The architecture is two-tier client-server and it applies the device/property model, where equipment is perceived as an device having properties that can be read, set and subscribed to. Also CMW provides a narrow interface to FESA servers in a similar way as IFC. The device's properties are controlled by synchronous and asynchronous get and set calls and subscription (monitor on and monitor off methods).

Data is transported in a similar way as in IFC, by using data container which is a map of primitive values which can be either scalars or arrays. The values in container can be accessed by name and therefore the knowledge of the location of specific value within the container is not necessary.

IFC and FESA comparison

In the table below mapping of the methods is listed. Both interfaces differ significantly but due to the implementation details as explained in next chapter this didn't present a problem.

Table 1: Overview of Method Mapping between IFC and CMW

Meaning of method	**IFC method name**	**CMW method name**
synchronous read	read	(synchronous) get
synchronous write	write	(synchronous) set
synchronous action	call	/
asynchronous read	requestRead	(asynchronous) get
asynchronous write	requestWrite	(asynchronous) set
asynchronous action	requestCall	/
cancel asynchronous process	cancelRequest	/
repetitive read	connectRead	/
repetitive write	connectWrite	/
repetitive action	connectCall	/
cancel repetitive process	disconnect	/
monitor on		monitorOn
monitor off		monitorOff

In the table below mapping of data types supported by both systems are listed. Also here the difference is significant since CMW does not support unsigned data types.

Table 1: Overview of Data Type Mapping between IFC and CMW

Data type/ C++ type	**Supported by IFC/ C++ type**	**Supported by CMW/ C++ type**
boolean	yes / signed short	yes / bool
signed byte	yes / signed char	yes/ signed char
unsigned byte	yes / unsigned char	no
signed word	yes / signed short	yes / signed short
unsigned word	yes / unsigned short	no
signed long	yes / signed long	yes / signed long
unsigned long	yes / unsigned long	no
float	yes / float	yes / float
double	yes / double	yes / double
string	yes / std:string	yes / char*

GATEWAY

General

Gateway is used to transform FESA device servers to present GSI control system IFC devices. This is achieved by transforming CMW calls and data types to IFC calls and data types as shown on figure 2.

Gateway is implemented as an IFC device server, running in the GSI device manager environment, which implements a corresponding IFC property for each property of the FESA device. The IFC property simply calls the FESA property via synchronous CMW access, mapping the FESA data types to IFC data types. This has a neat advantage in that the complete IFC interface is already provided in the IFC device server framework and no specific coding was necessary to handle asynchronous and connected requests. Connection between IFC interface and FESA only uses synchronous access which greatly simplifies architecture of the gateway.

A default implementation of the IFC properties, representing the FESA properties in the IFC device severer, is generated from the formal FESA class design description. This includes mapping between IFC and FESA data types.

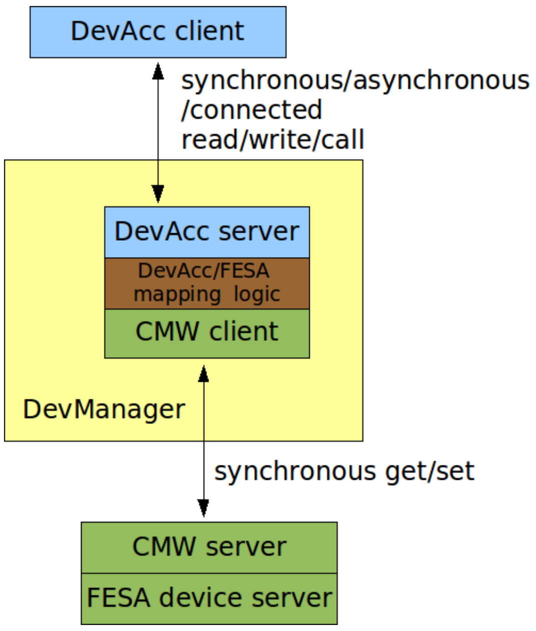

Figure 2: Gateway architecture

Mapping Logic

Default gateway implementation is sufficient for basic FESA to IFC properties transformation. This means each FESA property's data item (primitive value in data container - map) is transformed into appropriate IFC property's data item (primitive value in data container - vector). In a similar way each FESA property's filter item is transformed into appropriate IFC property's parameter item.

Gateway uses FESA class design XML file to generate rules for basic mapping logic described above. FESA class design XML file contains information on all FESA device server properties and also all data they contain. The order of data items defined in XML file will be preserved and used in IFC data container.

For complex transformations gateway provides extension points where arbitrary transformation between FESA and IFC properties can be defined. This is very useful when single IFC property is composed of several FESA properties or if several IFC properties require only small part of data provided by certain FESA property. During start up gateway checks if specific ICF property is implemented in extension point. If that is true then it's basic implementation is excluded in the process of generating properties.

Mapping of names of the properties is defined in special configuration file. In this file for each FESA property name corresponding IFC property name is defined. If certain property or whole file is missing gateway will only transform property names to upper case.

Gateway uses a set of configuration files for additional configuration. Currently only two are supported; name of the FESA server to which gateway should connect to and if gateway should be run in read only mode.

FURTHER DEVELOPMENT

For gateway to be really useful (is primary usage is to allow old IFC clients to be used with new FESA devices) a FESA class design guidelines will have to be written. The guidelines should include standard IFC properties (e.g. INIT, RESET, VERSION, STATUS, RESET) that will have to be implemented in each FESA class even if the functionality will not be available for particular device (dummy property will be used). This is necessary to avoid any failures in IFC clients. Guidelines should also contain agreement on how to include time stamp in the properties that perform data acquisition and how to implement IFC action which could be seen as a set without data.

Current implementation of the gateway is using FESA class design XML file to generate properties. This has a downside since the maintainer of gateway instances has to maintain also a set of additional XML files. To avoid this problem, gateway could use FESA database, from where all relevant data could be obtained and would thus require only the knowledge of the FESA class name and its version.

Before new alarm system is developed for FAIR project old alarm system could be used to collect alarms from new devices developed in FESA. To accomplish this gateway would create a special CMW monitor connections on relevant properties (e.g. Alarm, AlarmDetails, Status) as shown on the figure below.

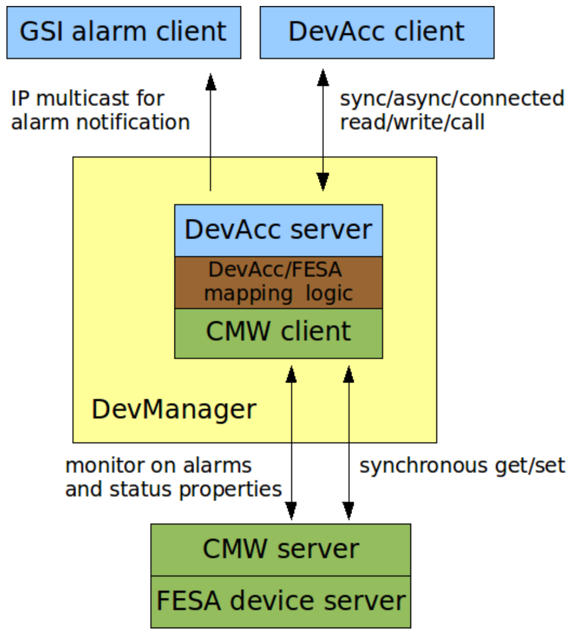
GSI alarm client
DevAcc client
IP multicast for alarm notification
sync/async/connected read/write/call
DevAcc server
DevAcc/FESA mapping logic
CMW client
DevManager
monitor on alarms and status properties
synchronous get/set
CMW server
FESA device server

Figure 2: Gateway layout with alarms

Gateway would maintain a list of active alarms collected from various properties on FESA device server and would fire alarm notifications via IP multicast as all other current GSI devices.

REFERENCES

[1] T. Hoffman, "FESA – The Front-End Software Architecture at FAIR", PCaPAC08
[2] I. Kriznar et al., "IFC/FESA gateway – project proposal and design", unreleased