VLT Central Control Software Tutorial:

DBL Database Loader

Extended CCS

Event Handling Tool-kit

G.Chiozzi
19/06/95

DBL Database Loader

The DBL Database Loader is the tool to be used for design and configuration of the real time database for all applications, both on Workstation side and on LCUs.

Purpose of the tool is:

- Provide an easier way of describing database structures, compared to RTAP point configuration files
- Allow for reuse of common database structures through object oriented concepts.
- Allow for integration of database branches distributed on Workstations and on LCUs

DBL Features

DBL is a preprocessor that convert a database description file in

"Branch Config File"

format into standard RTAP Point Config Files, that can be loaded both on Workstation and LCU databases.

The Branch Config File syntax provides the following features:

- Definition of a complete database branch in a single file
- Definition of Database Classes, with support for:
  - Inheritance
  - Overloading
  - Complex data types
- Support for all standard C preprocessor features (#include, #define, #if..#endif,...)
- Definition of Scan Links

DBL Future Development

- The July release of the DBL module will contain:
    - Support for the Alarm System
    - A library of classes for common point types used in applications. This library will be extended in the following releases to include new classes extracted from applications or developed on user request
- No major changes are foreseen for the preprocessor engine and Branch Config File syntax. Specific user requests for new features can be examined.
- The July release will also include an **Interactive Database Browser** to navigate in the class hierarchy. It will include (see figures):
    - Tree window to browse class hierarchy
    - Tree window to view point's internal structure
    - List window with list of defined classes
    - Definition window with display of one class

ECCS - Extended CCS

Extended CCS is a library written in C++ on top of CCS. It contains functions and simple classes and requires a minimal knowledge of C++ syntax. It uses C++ only as a "better C" and its usage does not imply the knowledge of object-oriented techniques. It runs only on Workstation.

Purpose of the library is:

- Provide easier usage of common CCS features
- Embed in functions or simple classes commonly used application patterns
- Provide base CCS classes for the development of truly object oriented applications
- Prevent common errors in applications
- Standardise produced code

The library is not (and does not want to be) a complete C++ replacement of CCS.
Only the commonly used parts of CCS are integrated in ECCS.
All standard CCS functions can be used whenever necessary.

ECCS Features

General features:

- Many commonly used CCS functions are overloaded with a simplified interface.
- Whenever possible, function parameters gets a reasonable default value
- Several macros and convenience functions implement commonly used application patterns.

Error handling:

- Error handling has been simplified with the assumption that an application uses only one “error stack”
- The **eccsERROR** class makes easier handling error stack structures
- The **eccsERROR_CLASS** class provides basic error handling features when writing new classes and object-oriented applications
- Macros and convenience functions implement standard error handling strategies.

ECCS Features (cont.)

Database access:

- **dbRead()** and **dbWrite()** overloaded functions provides a much easier access to database attributes
- Other functions and an **eccsDB_ATTR** class are under development and test.

Message system:

- **msgMESSAGE()** class encapsulate the CCS message system and makes much easier to receive, parse and send messages and commands.
- **msgRAW_MESSAGE()** is a base class to be used to implement customized handling of messages.

ECCS Features (cont.)

Event system:

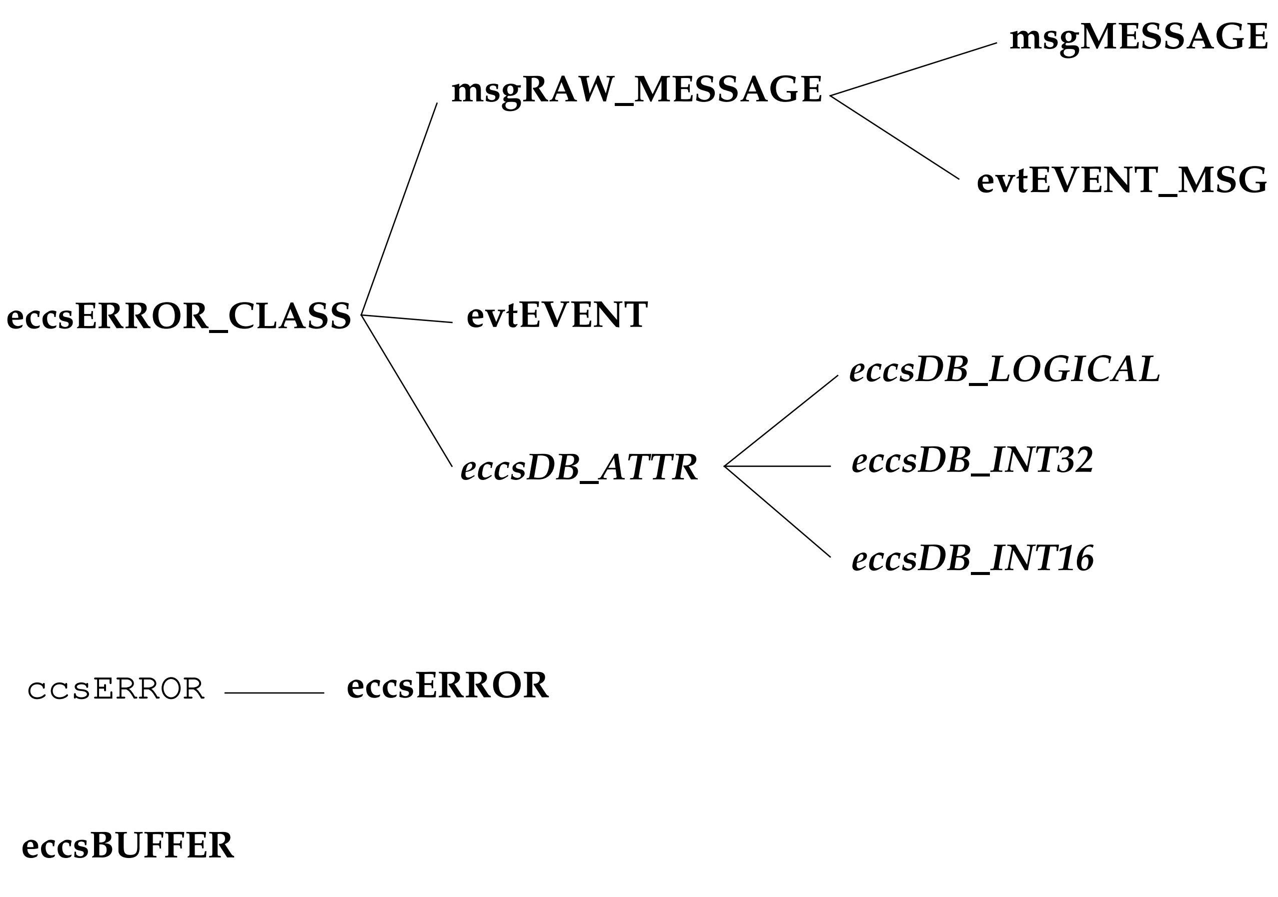
- **evtEVENT** class encapsulate handling of CCS events
- **evtEVENT_MSG** class is used to handle incoming Event Messages.

Fast prototyping and test tools:

- A set of functions and macros are provided to make easier debugging and writing of test programs

ECCS class tree

This diagram contains the tree of classes in ECCS module:



ECCS Future development

ECCS module will grow incorporating new functions based on:

- User's contributions
- User's requests

The preferred way is to receive user contributions and, after verification that they are conforming to standards and generally useful, integrate them in the module.

The July '95 release will include new feature for:

- Database access
- Debugging and test tools

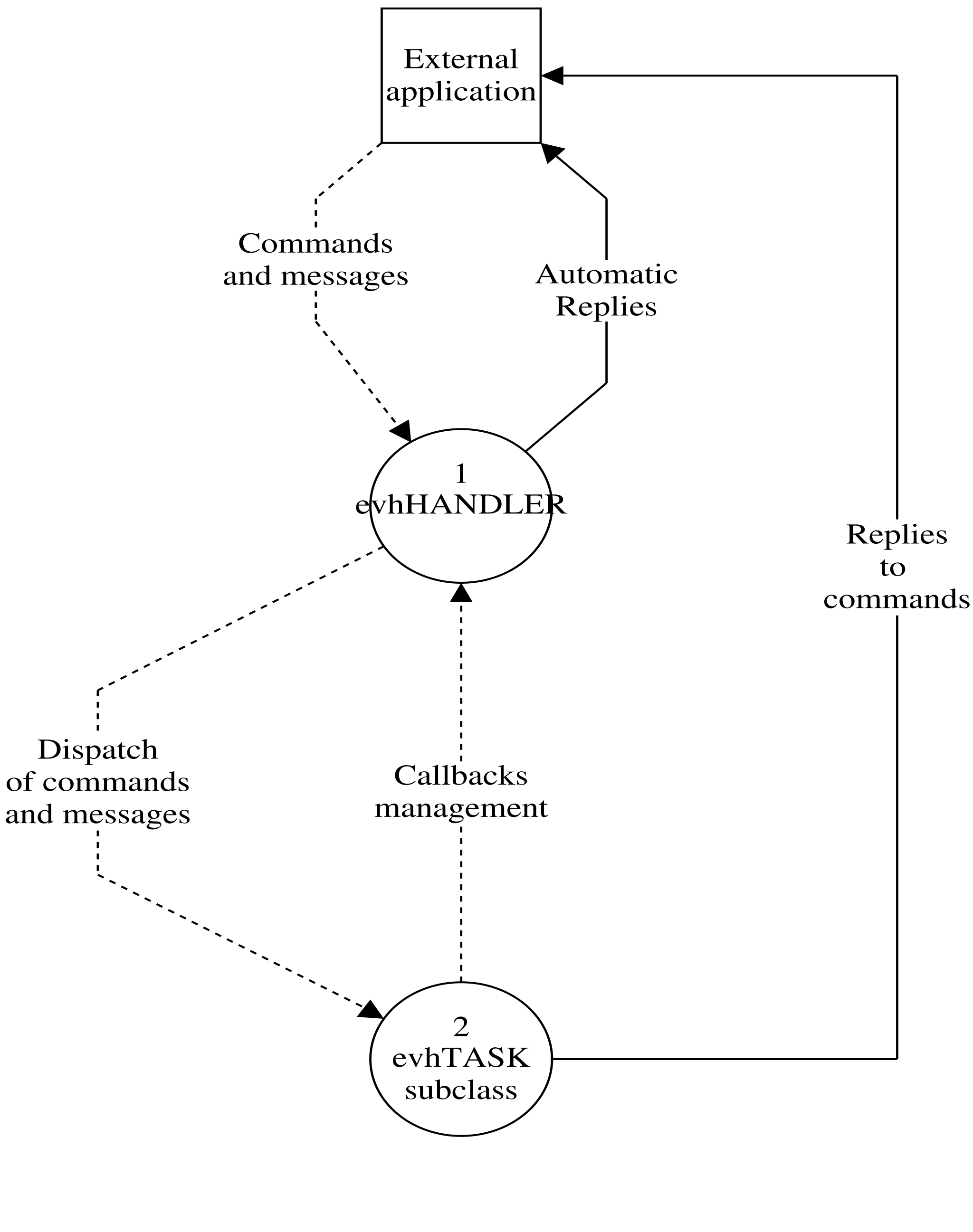
EVH - Event Handling Toolkit

The Event Handling Toolkit is a set of C++ classes to develop **event driven applications** on workstations.
Development of applications requires a knowledge of concepts for event driven programs and object-oriented programming techniques.

Purpose of EVH is:

- Provide a framework to implement easily workstation applications that receive and send commands and messages.
- Implement in a standard way common application patterns
- Prevent common errors in applications
- Support object-oriented design and development
- Provide tools for testing and debugging of applications.

EVH Application Structure

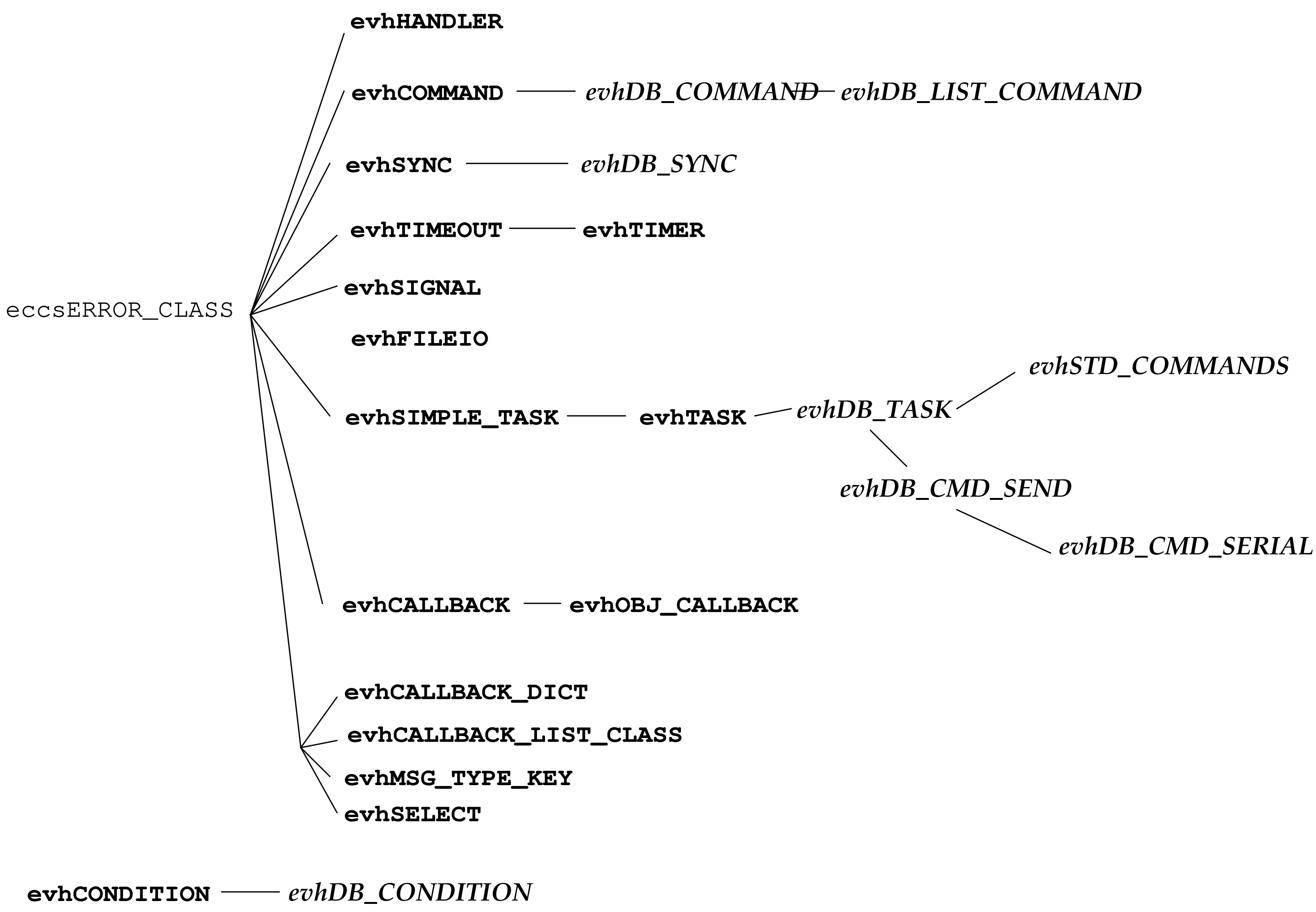


EVH Classes

- **Fundamental building blocks**
  - evhHANDLER
  - evhSIMPLE_TASK, evhTASK
- **Basic classes**
  - evhCOMMAND
  - evhSYNC
  - evhTIMEOUT
  - evhTIMER
- **Special classes**
  - evhFILEIO
  - evhSIGNAL
- **Support classes**
  - evhCALLBACK, evhOBJ_CALLBACK
  - evhCONDITION
  - evhMSG_TYPE_KEY
  - evhSELECT

EVH class tree

The following diagram shows the classes distributed with the current version of EVH and, in italics, classes already developed and under test.



EVH debugging tools

To help testing and debugging of applications, EVH provides:

- Logging features embedded in the framework
- Testing base classes
- Skeletons for testing applications
- The **evhDummy** program to emulate dynamical behaviour of clients of a program under test.

These feature, associated with what provided by ECCS, help to write standard modular and automatic test procedures for any application (workstation and even LCU based).

EVH Future development

EVH module will grow incorporating new functions based on:

- User's contributions
- User's requests

The preferred way is to receive user contributions and, after verification that they are conforming to standards and generally useful, integrate them in the module.

Several new classes have already been developed and are under test. They will be included in the July '95 release.

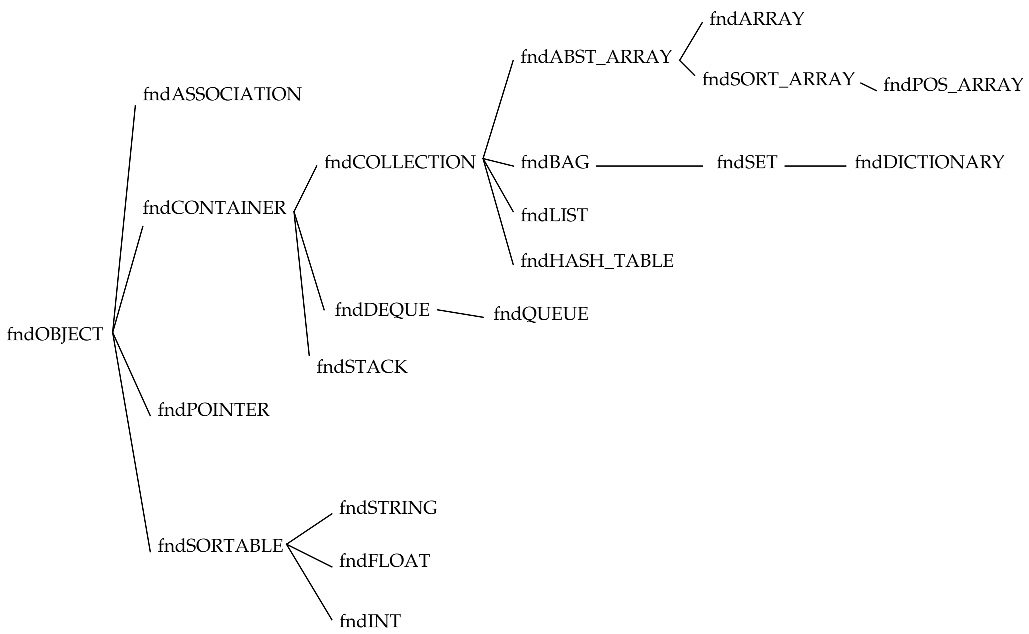
Many of these classes are related to database access and have a twin Database Loader class.

This is a general technique to be used to implement data structures stored on the database and methods in the form of twin C++ classes.

EVH companion library: Foundation Classes

A library of foundation classes **(fnd module)** is distributed with EVH.
It provides basic classes for common data structure handling (dynamic arrays, lists, stacks...).
It is now distributed without documentation. A user manual and man pages will be distributed in a next release.

The library provides the following classes:



Conclusion

The 3 modules:

DBL, **ECCS** and **EVH**

provide object-oriented concepts, at different levels, to make easier the design and development of VLT applications, mainly on the workstation's side.

The database and C++ class libraries are supposed to grow in the following releases on the base of user requirements and preferably incorporating user's contributions.

This is intended to improve reuse of (useful) code, warranting maintenance and uniformity of interface.