HRE – LOGGING - OVERVIEW

Revision history

2018-08-03	Robin Lamacraft	Original draft
2018-08-08	Don Ferguson	Tidy-up edits throughout
2018-10-16	Rod Thompson	Merge Commit-Logging into a wider Logging Overview
2018-10-27	Rod Thompson	Amend SCOPE Add APPENDIX 1 – Logging functions
2019-02-07	Rod Thompson	Add new section headings 1) GENERAL LOGGING 2) ERROR LOGGING Add definitions Add ERROR LOGGING Example – Appendix 2
2019-04-05	Rod Thompson	Addition to General Logging
2019-04-06	Don Ferguson	Minor edits and corrections to CL format table description
2019-04-11	Rod Thompson	Reformat – Add & Differentiate titles Revise wording for General Log files Add File Naming Convention references

GENERAL

SCOPE

Logging is a means of tracking events in the use of HRE.
Log files are created that may be inspected by the User.
Logging in HRE is intended to serve two purposes:

1) To aid investigation when some unexpected action occurs in software operation, facilitated by inspection of the Log files
2) To provide an 'undo' functionality – to reverse a recent action.

Logging, as presently defined in HRE specifications involves areas such as Commit-Logging, Errors, Warnings and Messages.

In keeping with the general principle adopted in the HRE design, functions not wanted by the User may be turned OFF (excluding Commit Logging).

REFERENCES

The following specifications are relevant.

1. *05.20 GUI_Logging* (to display log file entries)
2. *07.11 BR_Logging* (to perform logging activities).

COMMIT LOGGING

As an HRE Project Database is edited, HRE can record the sequence of changes in a form that:

- The user can view to understand who made what change when. These records are bundled together on the basis of the group of record changes that are made within one database Commit action
- The commit logging records are separate tables in the HRE database that will need to be managed to avoid their size getting too large. Some tools are provided to purge log records on different criteria

- The commit logs are likely to be very useful in understanding why the user ended up with an unexpected result and hence will be an aid to understanding and removing bugs
- The log records will allow the user to undo actions in reverse sequence to recover a corrupted database when possible
- Allows purging of Log entries (manually or through a software-controlled process).

This functionality cannot be turned off.

THE COMMIT LOGGING DATABASE STRUCTURE

There are 2 versions of each database table:

1. The tables that hold the active application control and user-entered data (as described in HRE specification documents). These tables are complete in themselves and are used when retrieving data values or composing output
2. The second set of tables with names prefixed with CL_ (for Commit Logging) have:
    a. their own unique PID identifier
    b. a set of fields which is an exact copy of the fields in the table being logged.

EDITING A GROUP OF TABLES TO CONSISTENTLY UPDATE THE DATABASE

Each proposed change to the database, which may involve making complementary changes to several database tables, has to be considered as the only method by which the database is modified:

1. The list of active tables that are involved in this update must be determined and the updating process must gain exclusive access to these records while the update proceeds
2. The whole process must be controlled by a singular SQL COMMIT action. That Commit action assigns a unique ID, the user requesting the change, and the time stamp of the start of the change
3. Once the record to represent this Commit action is created, then a series of records that contain the old data values of the records about to be overwritten are copied into equivalent CL_named tables of the same name as the original(s)
4. On completion of the old value copy operation, the revised values are stored in the live record
5. When this is successfully completed, the action completion time is saved in the Commit Identified record in the commit log
6. Finally, the records which were set with exclusive access are released.

USING THE COMMIT LOG

- A view of the most recent Commit action is presented at the top of a list of actions
- Each Commit can be undone
- A range of Commit action logs can be selected and printed or saved to a file.

GENERAL LOGGING

A record of general activities within the application to facilitate identification of issues arising from operation. It is anticipated that early logs will be extensive, and that as confidence in operation of the software grows, so logging entries will be reduce in number.

General Logging files are established for:

- the Project (Project Log) – residing on the Server with the Project files
- the User (User Log) – residing on the Client computer, with the User files.

Backups of the Log files are created:

- Project Log file – when the Project files are backed up

- User Log files – when the UserAUX file is backed up.

Refer to 03.29 Overview – Backup

Addition of records to the relevant Log files are detailed in the relevant specifications where the records are created as a consequence of actions.

FILE NAMING CONVENTIONS

Refer to 03.22 Overview – File Naming Conventions

ERROR LOGGING

If errors occur in the operation of HRE, and if these are non-fatal to the operation of the application, then it is likely that they will be identified and notified to the User via the Messaging system.

It is essential that these errors be logged, to ensure that they are either dealt with by the User, or where as a consequence of software issues, that they are then dealt with by the developer.

Error Logging files are established for:

- The HRE Server
- The HRE Client.

FILE NAMING CONVENTIONS

Refer to 03.22 Overview – File Naming Conventions

AUXILIARY FILE LOGGING

User and **Database** Auxiliary files are used outside of the H2 database within HRE, to facilitate accessibility without the need to open databases to gather such data.

Changes to these files occur as access to Projects changes, or where changes occur in hardware configurations which impact operation of HRE.

A log file is created of the changes made to the AUX files; means are provided to investigate the contents of the log files, and to purge log file entries.

Specifications yet to be drafted

Also refer to *03.32 Overview - Auxiliary (Non-Database) Files*.

CLIENT – SERVER COMMUNICATION PERFORMANCE LOG

Questions have been raised about the value of logging detail about Client-Server communications.
Of relevance are communications failures, low bandwidth, timeouts, etc.
Further consideration is required.

Specifications yet to be drafted.

APPENDICES

APPENDIX 1 -Logging Functions

The following list includes some of the *Events or Functions* that will be logged in HRE, within the General Log files.

- HRE Client start
- HRE Client close
- User
- Hardware change - display
- Edit App Settings

- Server login
- Server logoff

- Project Open
- Project Close
- Project Backup
- Project Restore
- Project CopyAs
- Project Rename
- Project Delete
- Project New
- Project Setting Changes
- Add Viewpoint
- Remove Viewpoint
- Create Viewpoint
- Add DATA window (to Viewpoint)
- Configure Data window
- Edit Viewpoint

- Move/Resize Program Window

- Create Name Style
- Create Location Style

- Warning Message
- Error Message
- Message from Server
- Acknowledgement (C to S)
- Message from Colleague
- Message to Colleague

APPENDIX 2 -Error Logging Functions

The following list includes some of the *Errors* that will be logged in HRE:

- This Remote Server is not on-line
- The project cannot be located on this Server.