BR_Setting – Setting Services

Revision history

2017-03-24	Robin Lamacraft	Original draft
2018-03-29	John Lucas	Modified Scope, added Notes, other minor changes
2018-04-06	John Lucas	Revised Scope wording, 1st paragraphs
2018-06-19	Rod Thompson	Replace GUI_Notification with GUI_Message Patterns Revise REQUIRED SERVICES - Dependencies
2018-07-12	Rod Thompson	Replace XML files with JSON Auxiliary

SCOPE

This module must be used to define and maintain settings and default values for a diverse array of data types and devices. These values are used in memory, but are stored in three places:

1. JSON files on the client computer – these are User-specific
2. JSON files on the server computer – these may be server and Project-specific
3. Table values within the project H2 database – these are Project-specific and user-specific. This collection of settings is only available after successfully opening the database and may overwrite some settings in memory that were derived from the JSON files.

Entries in locations are indexed by the ID of the User, meaning that settings can have personal preference. These settings are stored as a hierarchy of values to the depth required to service the details of that sub-tree.

- Each setting is defined as User-related or Project-Related
- Each setting value has a nominated format (which may be specified for some values) and validation rules. Each setting has a default value
- BR_Setting is used by the GUI_AppSetting module which is opened from the main Tools > App > Settings menu entry
- BR_Setting is used by the GUI_ProjectSetting module which is opened from the main Tools > Project > Settings menu entry
- Many other modules also use it to lookup settings
- GUI_Setting displays the settings. The methods in BR_Setting operate on the settings data as driven by requests from GUI_Setting to create, edit individual setting values
- The setting Syntax and Validation rules are kept in the Setting Rules database and the Project Setting Rules database
- BR_Setting is also used by other modules to retrieve specific settings values
- These are values that the User can change.

ITEMS THAT HAVE SETTINGS AND DEFAULTS (incomplete list)

Viewpoints, Panels, Projects, Logging, Tags, Name Styles, Keyboard, Monitor, Pointer, Printer, Languages, Historical Date formats, field input sequences, and many more.

ACTIONS

The fundamental operations are:

1. Define a new Setting at the desired place in the setting hierarchy (each Setting has a unique HRE_ID) This is an admin or installation (HRE or plug-in) action
2. Create a record for the setting in the Client Setting or Project Setting Database
3. Retrieve a setting by its HRE_ID
4. Perform editing, syntax checking and value validation of entered values.

NOTES

1. The process of recording/setting a setting is separate from the retrieval of a setting value. Although both actions 2 and 3 will be done by the BR_Setting module, the setting action can only be done from (usually) one GUI screen for that setting, whereas the retrieval of a value is likely to be done at the request of another BR module or a number of GUI screens
2. A dictionary of setting value slots (and IDs) needs to be created with their purpose, value type and possible value set/range. This should say where they are stored when and how they are updated and the list of modules that use or display these values. Some settings will be embedded within the Project Database while others may be stored in the User and Project JSON files.

USED BY
1. GUI_AppSetting and GUI_ProjectSetting (for management of the setting databases)
2. Any module which wants to access the current or default value of a setting.

AUXILIARY DATA USED
- User Settings JSON file (User Auxiliary file)
- Project Settings JSON file (Project Database Auxiliary file)
- Project H2 database
    a. Table 303 – Viewpoint detail
    b. Table 305 – Screen detail (window)
    c. Table 126 – Project detail
    d. Table 129– Logging
    e. Table 153 – Tags
    f. Table 162– Name styles
    g. Table 202 – Languages
    h. Table 173, 175- Dates

REQUIRED SERVICES – Dependencies

First-Order Dependencies	Second-Order Dependencies	Higher-Order Dependencies
BR_AppData		
BR_UserData		
07.02 BR_EntityLink	Needed	

APPLICATION PROGRAMMING INTERFACE (API)
1. Needs details

WARNING CONDITIONS
1. Need details of the condition that raised the warning, example message and possible next steps. (GUI_Message Patterns to be used)

ERROR CONDITIONS
1. Need to record the condition that raised the error, example message and possible next steps. (GUI_Message Patterns to be used)