Using UCF System Variables

Richard Todd
Test Engineering Services
03.15.2023

1. Description

System variables provide an alternative to hard-coded system configuration values.
The following types of variables are supported

a) Global variables that are common to all profiles

b) Local variables that are common to a specific profile group

Local variables will always override global variables of the same name.

1.1. Data Locations

System variables are stored in local or global sqlite databases and are configured using the ‘DB Browser for Sqlite’ application.

Locations

Database Type	Database Location	Platform
Local	*~/ucfroot/opt/<Profile Family Name>/ExtSystemIO.sdb*	Linux
	C:\ucfroot\opt\<Profile Family Name>\ExtSystemIO.sdb	Windows
Global	*~/ucfroot/bin/<Release Name>/IOConfig/BaseSystemIO.sdb*	Linux
	C:\ucfroot\bin\<Release Name>\IOConfig\BaseSystemIO.sdb	Windows

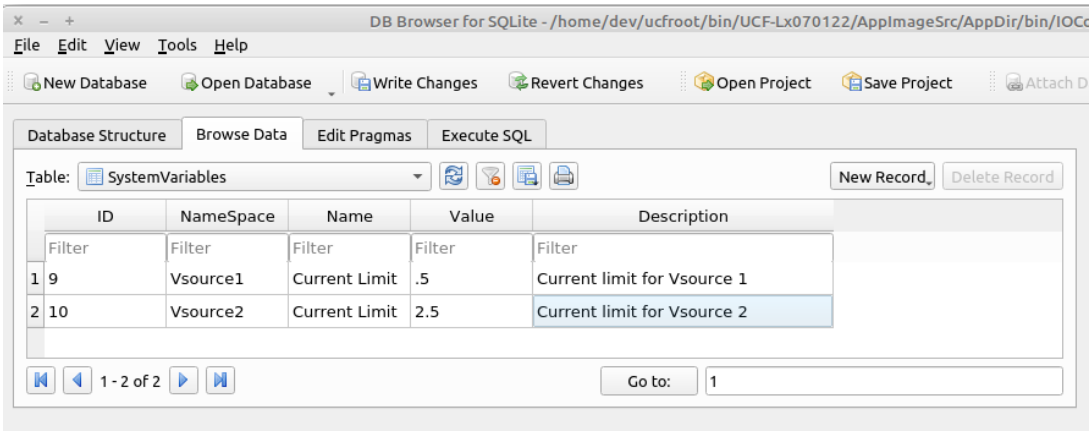
a) Local database values are unique to a specific profile family and will override values in global databases.

b) Global database values will be used for all profiles that do not have overriding values in a local database.

2. Configuring System Variables

System variables are created and updated using the sqlitebrowser utility.

- Available at: https://sqlitebrowser.org/



3. Reading System Variables

The value(s) for system variables are read programmatically using the '*GetSystemVariable(...)*' functions defined in iStdIOChannelSystem.hpp.

Example 1:

```
bool status=TheSystem->GetSystemVariable(..);
/*  TheSystem is an iStdIOChannelSystem*
    1.  status will be true if the variable is found, and false if not
    */
```

Example 2:

```
int TplModule::TestSystemVariables(CmdParam msg) {

        std::string ns, name, value, desc;
        char cvalue[256];
        memset(cvalue,0,256);

        TokenString params(msg,',');
        if (params.GetTokenCount() > 3) {

            params.GetToken(1, ns);
                params.GetToken(2, name);
                params.GetToken(3, value);
                params.GetToken(4, desc);

                TheSystem->GetSystemVariable(ns.c_str(), name.c_str(),cvalue,256);
        }

        return(1);
}
```

3.1. TokenString Methods

The following TokenString methods are useful for reading multi-type system variables with default values.

std::string **GetStrToken**(unsigned index,std::string default_value="");

IOString **GetIOStrToken**(unsigned index,std::string default_value="");

bool **GetToken**(unsigned index,std::string &value, std::string default_value="");

bool **GetToken**(unsigned index,unsigned &value,unsigned default_value=std::numeric_limits<unsigned>::min());

bool **GetToken**(unsigned index,int &value,int default_value=std::numeric_limits<int>::min());

bool **GetToken**(unsigned index,bool &value,bool default_value=false);

bool **GetToken**(unsigned index,double &value,double default_value=std::numeric_limits<double>::min());

```
/*  get the specified token, and return the value as an std::string using 'default_value'
        if the token is not found
        */
```