Fred report engine reference (API v2)

1 Copyright

2 Content / Target audience

This document is mainly targeting fred users who want to build their own report templates and contains detailed descriptions of the reporting engine and all constants, data types and functions that are exported by fred to be used in report templates.

3 Fred, report engine. What's this all about?

Forensic Registry EDitor (fred) is a cross-platform M$ Windows registry hive viewer / editor including special features useful during forensic analysis. One of this features is the report engine. It allows users to write their own report templates to extract often used information from registry hives and represent it in a nicely formated report. This may for example be such simple information like the used Windows version or more complex information like all user accounts. Report templates are written in a script language that implements all the built-in objects and properties defined in the ECMA-262 standard (http://www.ecma-international.org/publications/standards/Ecma-262.htm). Another popular implementation of this standard is JavaScript. Using functions that are exported by fred, report templates gather the needed informations from the currently loaded registry hive and generate a HTML page which is then shown to the user.

4 Report template storage locations and file naming conventions

There are two (2) default locations where report templates may be stored. One contains report templates that are available to all users of a system. Any report templates that are distributed along with the official fred builds will be stored there. The other location is user specific. All reports that are stored there will only be visible to that user. The following table lists both storage locations:

OS	**System storage location**	**User storage location**[1]
All Unix like systems	/usr/[local/]share/fred/report_templates	~/.fred/report_templates
M$ Windows	<Application install dir>\report_templates	<Home folder>\.fred\report_templates

Report template file names should use the following naming convention:

<CATEGORY>_<Report name>.qs

Files not having a ".qs" file extension will not be considered valid report templates.

[1] This folder will be created when you run fred for the first time.

5 Basic report template structure

A basic report template must implement the following two (2) functions:

5.1 fred_report_info

This function is called by fred to get informations about the report template. It's only action is to create and return an associative array containing the following key/value pairs:

Key name	Description
report_cat	String containing the report template's category name.
report_name	String containing the report template's name.
report_author	String containing the report template's author.
report_desc	String containing the report template's description.
fred_api	Integer informing fred about the minimum API version this report template is compatible with.
hive	String containing the hive for which this report template was designed. This may be “UNKNOWN”, “SYSTEM”, “SOFTWARE”, “SAM”, “SECURITY” or “NTUSER”.

5.2 fred_report_html

This function generates the actual report in HTML.

6 Hello world

The following code generates a very basic report that displays “Hello World”. It shows you the basic structure of a report template. Use your favorite text editor, insert the following code and save it in your user's report_template folder under the name “MISC_HelloWorld.qs”. After restarting fred and opening a registry hive, it should be available under Reports → MISC → HelloWorld.

```
function fred_report_info() {
  var info={report_cat    : "MISC",
            report_name   : "HelloWorld",
            report_author : "Gillen Daniel",
            report_desc   : "Simple HelloWorld demo report",
            fred_api      : 2,
            hive          : "UNKNOWN"
  };
  return info;
}

function fred_report_html() {
  println("    <h2>Hello World</h2>");
}
```

println is one of the functions that fred exports. It simply adds a line containing the specified text to an internal buffer. This buffer is then interpreted by fred as a HTML page and shown to the user. You are free to use any HTML 4 tags in your report and to use Cascading Style Sheets (CSS) to further format the report.

7 Fred constants

The following constants are available in all report templates:

Name	Type	Description
ENGINE_API_VERSION	Integer	Fred report engine's API version.
HIVE_FILE	String	Path and filename of currently opened hive file.

8 Fred data types

In order to be able to work with the mainly binary data stored inside registry hives, fred extends the ECMA-262 standard with the following data types.

8.1 RegistryKeyValue

This data type is a simple structure representing a registry key value and contains three (3) fields:

- *type*: An integer containing the key's value type.
- *length*: An integer containing the key's value length in bytes.
- *value*: A ByteArray containing the key's value.

The value type constants are not exported by fred to any report templates. If you need type checking, pass the type value to the RegistryKeyTypeToString() function and use the resulting string.

It is not possible to create your own RegistryKeyValue values inside a report template!

8.2 ByteArray

This data type represents an array of single 8bit (=1byte) values. Unlike RegistryKeyValue, you may create your own byte arrays inside a report template. The following methods are exported:

8.2.1 append / appendByte

Syntax: ByteArray.append(<other_ba>);
Syntax: ByteArray.appendByte(<byte>);
Return value: Nothing.
Description: Append another byte array / byte.
Example: ba.append(another_ba);
ba.appendByte(0xf7);

8.2.2 chop / remove / truncate

Syntax: ByteArray.chop(<len>);
Syntax: ByteArray.remove(<pos>,<len>);
Syntax: ByteArray.truncate(<pos>);
Return value: Nothing.
Description: Chops (removes) len bytes from the end of the byte array. / Removes len bytes from the byte array, starting at index position pos. If pos is out of range, nothing happens. If pos is valid, but pos + len is larger than the size of the byte array, the array is truncated at position pos. / Truncates the byte array at index position pos. If pos is beyond the end of the byte array, nothing happens.
Example: ba.chop(3);
ba.remove(2,3);
ba.truncate(7);

8.2.3 equals

Syntax: ByteArray.equals(<other_ba>);
Return value: True or false.
Description: Compares two byte arrays.
Example: if(ba.equals(other_ba)) { println(“ByteArrays are equal”); }

8.2.4 left / right

Syntax: ByteArray.left(<len>);
Syntax: ByteArray.right(<len>);
Return value: New byte array containing leftmost / rightmost len bytes of this byte array.
Description: Extracts leftmost / rightmost len bytes of this byte array. If len is greater than size(), the entire byte array is returned.
Example: new_ba=ba.left(3);

8.2.5 mid

Syntax: ByteArray.mid(<pos>,[<len>]);
Return value: Byte array containing len bytes from this byte array, starting at position pos.
Description: Extracts len bytes from this byte array, starting at position pos. If len is -1 (the default), or pos + len >= size(), returns a byte array containing all bytes starting at position pos until the end of the byte array.
Example: new_ba=ba.mid(3,5);

8.2.6 size

Syntax: ByteArray.size();
Return value: Integer representing the amount of bytes in this byte array.
Description: Returns the number of bytes in this byte array. The last byte in the byte array is at position size()-1.
Example: println("Byte array size: ",ba.size());

8.2.7 toBase64 / toLatin1String / toLower / toUpper

Syntax: ByteArray.toBase64();
Syntax: ByteArray.toLatin1String();
Syntax: ByteArray.toLower();
Syntax: ByteArray.toUpper();
Return value: New byte array / string (in case of toLatin1String) converted according to the called method.
Description: Returns a copy of the byte array, encoded as Base64, converted to lowercase / uppercase or a latin1 string.
Example: new_ba=ba.toBase64();
println("Byte array contents: ",ba.toLatin1String());

8.2.8 trimmed

Syntax: ByteArray.trimmed();
Return value: New byte array that has whitespace removed from the start and the end.
Description: Returns a byte array that has whitespace removed from the start and the end. Whitespace means any character for which the standard C++ isspace() function returns true. This includes the ASCII characters '\t', '\n', '\v', '\f', '\r', and ' '.
Example: new_ba=ba.trimmed();

9 Fred functions

The following functions are usable in all report templates to get and convert data from registry hives.

9.1 print / println

Syntax: print([<string1>, [<string2>, …]]);
Syntax: println([<string1>, [<string2>, …]]);
Return value: Nothing.
Description: Add a string (print) or a line (println) to the internal buffer that is later converted to a HTML page and shown to the user.
Example: print("Hello World");
println("This results in a ", "line with appended new-line char");

9.2 GetRegistryNodes

Syntax: GetRegistryNodes(<path>);
Return value: Array containing all node names under the given path or undefined.
Description: Fetches an array containing all direct child nodes of a specified path.
Example: var nodes=GetRegistryNodes("\\some\\registry\\path");

9.3 GetRegistryKeys

Syntax: GetRegistryKeys(<path>);
Return value: Array containing the key names or undefined.
Description: Fetches an array containing all child keys of a specified path.
Example: var keys=GetRegistryKeys("\\some\\registry\\path");

9.4 GetRegistryKeyValue

Syntax: RegistryKeyValue GetRegistryKeyValue(<path>, <key>);
Return value: RegistryKeyValue structure or undefined.
Description: Fetches the value, it's byte length and value type of a specified key located under a

specified path.
Example: var key_value=GetRegistryKeyValue("\\some\\registry\\path", "MyKey");

9.5 GetRegistryNodeModTime

Syntax: GetRegistryNodeModTime(<path>);
Return value: String containing node's last modification time or undefined.
Description: Returns the date / time ("yyyy/mm/dd hh/mm/ss") the node was last modified.
Example: println("Node '\\some\\path' last modified ", GetRegistryNodeModTime("\\some\\path"));

9.6 RegistryKeyValueToString

Syntax: RegistryKeyValueToString(<value>, <value_type>);
Return value: String containing the converted value or undefined.
Description: Converts the give registry key value to a string according to it's value type.
Example: var key_value=GetRegistryKeyValue("\\some\\registry\\path", "MyKey");

```
if(typeof key_value !== 'undefined') {
    print("MyKey value: ");
    println(RegistryKeyValueToString(key_value.value, key_value.type));
}
```

9.7 RegistryKeyValueToStringList

Syntax: RegistryKeyValueToStringList(<value>, <value_type>);
Return value: Array containing the converted value or undefined.
Description: Converts the give registry key value to an array of strings. Should only be used to convert values of type "REG_MULTI_SZ".
Example: var key_value=GetRegistryKeyValue("\\some\\registry\\path", "MyKey");

```
if(typeof key_value !== 'undefined') {
    print("MyKey values: ");
    var strings=RegistryKeyValueToStringList(key_value.value,key_value.type);
    for(var i=0;i<strings.length;i++) {
       println("         ",strings[i],"<br />");
    }
}
```

9.8 RegistryKeyValueToVariant

Syntax: RegistryKeyValueToVariant(<value>, <format>, [<offset>, [<length>,[<endianness>]]]);
Return value: String containing the converted value or undefined.
Description: Converts the give registry key value to a string according to the given format. Optionally, only length bytes starting from offset are converted and returned. The last optional parameter specifies the endianness that should be used for decoding. 1 = little endian (default) / 0 = big endian. Available formats are:

Format string	Description
"int8"	Interprets and formats value as signed 8bit (1byte) integer.
"uint8"	Interprets and formats value as unsigned 8bit (1byte) integer.
"int16"	Interprets and formats value as signed 16bit (2byte) integer.
"uint16"	Interprets and formats value as unsigned 16bit (2byte) integer.
"int32"	Interprets and formats value as signed 32bit (4byte) integer.
"uint32"	Interprets and formats value as unsigned 32bit (4byte) integer.
"int64"	Interprets and formats value as signed 64bit (8byte) integer.
"uint64"	Interprets and formats value as unsigned 64bit (8byte) integer.
"unixtime"	Interprets and formats value as 32bit (4byte) Unix time.
"filetime"	Interprets and formats value as 64bit (8byte) Windows filetime.

“ascii”	Interprets and formats value as ASCII string. Warning: Value must be NULL-terminated or use length parameter!
“utf16”	Interprets and formats value as UTF-16 string. Warning: Value must be NULL-terminated or use length parameter!

Example:
```
var key_value=GetRegistryKeyValue(“\\some\\registry\\path”, “MyKey”);
if(typeof key_value !== 'undefined') {
    print(“Value interpreted as UTF-16 string: “);
    println(RegistryKeyValueToVariant(key_value.value, ”utf16”));
}
```

9.9 RegistryKeyTypeToString

Syntax: RegistryKeyTypeToString(<value>, <value_type>);
Return value: String containing the converted value or undefined.
Description: Converts the give registry key type to it's string representation. The following types exist:

Type	Description
“REG_NONE”	A key without a value.
"REG_SZ"	A Windows string (Encoding is unknown, but often UTF16-LE. Aren't they funny at M$?)
"REG_EXPAND_SZ"	A Windows string that contains environment variables.
"REG_BINARY"	A binary blob.
"REG_DWORD"	A 32bit (4byte) little endian integer.
"REG_DWORD_BIG_ENDIAN"	A 32bit (4byte) big endian integer.
"REG_LINK"	A symbolic link to another part of the registry tree.
"REG_MULTI_SZ"	Multiple Windows strings. See http://blogs.msdn.com/oldnewthing/archive/2009/10/08/9904646.aspx
"REG_RESOURCE_LIST"	A resource list.
"REG_FULL_RESOURCE_DESC"	A resource descriptor.
"REG_RESOURCE_REQ_LIST"	A resource requirements list.
"REG_QWORD"	A 64bit (8byte) integer with unspecified endianness (Usually little endian, but who knows!).
“0xXXXXXXXX”	Unknown types will be formated as 32bit (4byte) hexadecimal value prepended with “0x”.

Example:
```
var key_value=GetRegistryKeyValue(“\\some\\registry\\path”, “MyKey”);
if(typeof key_value !== 'undefined') {
    println(“Value type: “, RegistryKeyTypeToString(key_value.type));
}
```

10 Examples

10.1 Windows installation date / time (SOFTWARE hive)

```
function fred_report_info() {
  var info={report_cat    : "SOFTWARE",
            report_name   : "WindowsInstallDate",
            report_author : "Gillen Daniel",
            report_desc   : "Report Windows installation date / time",
```

```
            fred_api      : 2,
            hive          : "SOFTWARE"
  };
  return info;
}

function fred_report_htm() {
  println("   <h2>Windows installation time / date</h2>");
  println("   <table style=\"margin-left:20px; font-size:12; white-space:nowrap\">");

  var val=GetRegistryKeyValue("\\Microsoft\\Windows NT\\CurrentVersion","InstallDate");
  print("      <tr><td>Install date:</td>");
  if(typeof val !== 'undefined') {
    println("<td>",RegistryKeyValueToVariant(val.value,"unixtime"),"</td>");
  } else {
    println("<td>unknown</td>");
  }

  println("   </table>");
}
```

10.2 Recent documents (NTUSER.DAT hive)

```
function fred_report_info() {
  var info={report_cat     : "NTUSER",
            report_name    : "RecentDocuments",
            report_author  : "Gillen Daniel",
            report_desc    : "Report recently used documents",
            fred_api       : 2,
            hive           : "NTUSER"
  };
  return info;
}

function fred_report_htm() {
  println("   <h2>Recent documents</h2>");
  println("   <p style=\"font-size:12\">");
  println("    <table style=\"margin-left:20px; font-size:12\">");

  // Get list of recent docs
  var recent_docs=
    GetRegistryKeyValue("\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\RecentDocs",
                        "MRUListEx");

  // Iterate over all recent docs
  var i=0;
  var runlist=RegistryKeyValueToVariant(recent_docs.value, "uint32",i);
  while(Number(runlist) != 0xffffffff) {
    var entry=
    GetRegistryKeyValue("\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\ RecentDocs",
                        runlist.toString(10));
    println("      <tr><td>", RegistryKeyValueToVariant(entry.value, "utf16", 0), "</td></tr>");
    i+=4;
    runlist=RegistryKeyValueToVariant(recent_docs.value, "uint32",i);
  }

  println("   </table>");
  println("  </p>");
}
```