

# **GT 4.2.1 C Common Libraries: Developer's Guide**

---

## **GT 4.2.1 C Common Libraries: Developer's Guide**

### **Introduction**

The C Common Libraries provide an abstraction layer for data types, libc system calls, and data structures used throughout the Globus Toolkit and useful for applications that use the Globus Toolkit.

---

# Table of Contents

1. Before you begin .....	1
1. Feature summary .....	1
2. Tested platforms .....	1
3. Backward compatibility summary .....	1
4. Technology dependencies .....	1
5. Security Considerations for C Common Libraries .....	1
2. Usage scenarios .....	2
3. Architecture and design overview .....	3
4. APIs .....	4
1. Component API .....	4
2. Internationalization Infrastructure .....	4
3. Developer Information .....	4
5. Environment variable interface .....	6
1. Environmental variables for C Common Libraries .....	6
6. Debugging .....	7
7. Troubleshooting .....	8
8. Related Documentation .....	9

---

# Chapter 1. Before you begin

## 1. Feature summary

Features new in release GT 4.2.1:

- `globus_range_list` abstraction added
- `globus_logging` abstraction added
- In this release we added `globus_options`. This is some common code for parsing options from the command line, environment variables, or configuration files.

## 2. Tested platforms

The C common libraries work on any platform on which the toolkit is supported. See [supported platforms](#).

## 3. Backward compatibility summary

API changes since GT version 4.0.x

- `globus_range_list` abstraction added
- `globus_logging` abstraction added

All of the GT 3.2 API is still functional in GT 4.2.1.

## 4. Technology dependencies

C Common Libraries only depend on the `globus_core` module.

## 5. Security Considerations for C Common Libraries

There are no security considerations for the C Common Libraries.

---

# Chapter 2. Usage scenarios

C Common libraries will need to be used if virtually any other toolkit component is used, since many data types are abstract and require the C common libraries to manipulate.

---

# Chapter 3. Architecture and design overview

Not available at this time.

---

# Chapter 4. APIs

## 1. Component API

See the [C API pages](#)<sup>1</sup> for other API documentation on globus\_common.

## 2. Internationalization Infrastructure

The Globus Toolkit C Common Library now has optional infrastructure support for internationalization, which is used by GridFTP and its dependencies (non-ws authorization/authentication and XIO).

This means that user-presented strings are wrapped in a lookup function, which, if the globus\_i18n module is installed and the GLOBUS\_I18N environment variable is set to "YES", will lookup the string in a resource bundle using ICU4C. If GLOBUS\_I18N is set to "NO", or the globus\_i18n module is not installed, or the string value cannot be found in the resource bundle, the default string (exactly what was being looked up) is returned.

## 3. Developer Information

There are two functions that are used for string lookup.

The first is the preferred function; you supply a module descriptor and the string that you want to look up:

```
globus_common_i18n_get_string(  
    globus_module_descriptor_t * module,  
    char * key);
```

The second function is used if you need to look up a particular locale. A NULL value for locale will look up from the default locale:

```
globus_common_i18n_get_string_by_key(  
    char * locale,  
    char * resource_name,  
    char * key);
```

Typically, one or more macros will be defined on a per-module basis that supply the module descriptor, to reduce clutter in the code. Within the toolkit, these are typically `___SL` where `___` are some mnemonic for the module in question (for example `_GCSL` is defined for `globus_common`).

The resource bundles used for the string lookups are created using ICU4C (see [IBM documentation on Resource Bundles](#))<sup>2</sup>.

Our resource bundles are very simple; they contain simply a set of keys and strings. The key is actually the string itself: it is hashed using the `globus_hashtable_string_hash` function, then converted to contain only invariant characters (`#!@[]^{}~` are converted to `'_'`). See `globus_i18n_resource_init.c` in the `globus_i18n` source for an example of creating keys.

---

<sup>1</sup> <http://www.globus.org/api/c-globus-4.2.1/>

<sup>2</sup> <http://oss.software.ibm.com/icu/userguide/ResourceManagement.html>

While resource bundles for the Globus Toolkit are not by default created as part of the build process nor distributed in our binary distributions, there is a tool distributed with the `globus_i18n` package that makes them simple to construct.

Invoking `globus-i18n-resource-create <module name>` from the top level directory of a built (or at least configured) source package will generate a resource bundle for that package which can then be moved to `$GLOBUS_LOCATION/share/i18n/`.

`globus-i18n-resource-create` is part of the `globus_i18n` package, and will be installed if `--enable-i18n` was given as a configure option to the installer. `globus-i18n-resource-create` uses `globus-i18n-resource-init` and `genrb` from ICU4C to create the resource bundles.

Resource bundles must be installed in `$GLOBUS_LOCATION/share/i18n/`.

---

# Chapter 5. Environment variable interface

## 1. Environmental variables for C Common Libraries

- `GLOBUS_ERROR_VERBOSE=1` can be set to enable verbose error messages.
- `GLOBUS_ERROR_OUTPUT=1` can be set to enable output of all errors (including some that should be ignored).

---

# Chapter 6. Debugging

General C debugging techniques apply when developing with the C common libraries.

---

# Chapter 7. Troubleshooting

There are no specific troubleshooting techniques for the C common libraries.

---

# Chapter 8. Related Documentation

See the [C API pages](#)<sup>1</sup> for more information about this component.

---

<sup>1</sup> <http://www.globus.org/api/c-globus-4.2.1/>