Replica Location Service RPC Protocol Description

January 6, 2003

In this document, we describe the simple wire-level remote procedure call protocol used for communication between an RLS client and server. This description will be of interest to those developing alternative RLS client APIs.

1 RPC Encoding Conventions

Method names, parameters and results are all encoded as null terminated strings. In the case where a list of results is returned (e.g., after a wildcard query of {lfn,pfn} mappings), the list is terminated with an empty string (i.e., the last two bytes are nulls, one to terminate the last string in the list, and a second null for the empty string).

Integer and floating point values are encoded as strings using sprintf(3) in the C API. When passed as parameters, date values are encoded as YYYYMMDDHHMMSS. When returned as a result, a date value is encoded as YYYY-MM-DD HH:MM:SS. (These are the formats used by MySQL by default.) Enumerated type (enum) values (e.g., globus_rls_attr_type_t) are converted to their integer value and sent encoded as an integer. Note that enum values begin at 0.

2 RPC Method Invocation

All RPC method invocations begin with the method name, a null byte, and the parameters, which are null terminated strings.

All results begin with an integer result code (written as a null terminated string), and if successful (result code = GLOBUS_RLS_SUCCESS = 0), and if the method returns results, then the results follow the result code as null terminated strings.

If an error occurs, the result code will be non-zero, and an error message will follow the result code.

The RPC is layered on top of Globus-IO and sets the authentication mode to:
GLOBUS_IO_SECURE_AUTHENTICATION_MODE_GSSAPI
and the authorization mode to:
GLOBUS_IO_SECURE_AUTHORIZATION_MODE_HOST.

3 RPC Methods

The methods documented below were current at the time this document was written. However, since the API continues to evolve, this document may be slightly out of date. The definitive version of the C API can be found in globus_rls_client.h. It should be straightforward to determine the wire protocol from the function declaration using the encoding rules specified here.

METHOD: admin
PARAM: globus_rls_admin_cmd_t cmd

METHOD: close

METHOD: get_configuration
PARAM: char *option
RESULT: char *option
char *value
... (may be multiple option values)

METHOD: lrc_add
PARAM: char *lfn
PARAM: char *pfn

METHOD: lrc_attr_add
PARAM: char *key
PARAM: globus_rls_obj_type_t objtype
PARAM: globus_rls_attr_type_t type
PARAM: char *attr_name
PARAM: char *attr_value (may be int, float, char * or date, if date
format should be YYYYMMDDHHMMSS)

METHOD: lrc_attr_create
PARAM: char *attr_name
PARAM: globus_rls_obj_type_t objtype
PARAM: globus_rls_attr_type_t type

METHOD: lrc_attr_delete
PARAM: char *attr_name
PARAM: globus_rls_obj_type_t objtype
PARAM: int clearvalues

METHOD: lrc_attr_get
PARAM: char *attr_name
PARAM: globus_rls_obj_type_t objtype
RESULT: char *attr_name
globus_rls_attr_type_t type
... (may be multiple attr_name, type pairs)

METHOD: lrc_attr_remove
 PARAM: char *key
 PARAM: attr_name
 PARAM: globus_rls_obj_type_t objtype

METHOD: lrc_attr_search
 PARAM: char *attr_name
 PARAM: globus_rls_obj_type_t objtype
 PARAM: globus_rls_attr_op_t operator
 PARAM: char *operand1 (may be int, float, string or date (YYYYMMDDHHMMSS))
 PARAM: char *operand2 (may be int, float, string or date (YYYYMMDDHHMMSS))
 RESULT: char *key
        char *type
        char *attr_value (int, float, string or date (YYYY-MM-DD HH:MM:SS))
        ... (may be multiple key,type,attr_value tuples)

METHOD: lrc_attr_value_get
 PARAM: char *key
 PARAM: char *name
 PARAM: globus_rls_obj_type_t objtype
 RESULT: char *attr_name
        globus_rls_attr_type_t type
        int attr_value (int, float, string or date (YYYY-MM-DD HH:MM:SS))
        ... (may be multiple attr_name,type,attr_value tuples)

METHOD: lrc_clear

METHOD: lrc_create
 PARAM: char *lfn
 PARAM: char *pfn

METHOD: lrc_delete
 PARAM: char *lfn
 PARAM: char *pfn

METHOD: lrc_exists
 PARAM: char *key
 PARAM: globus_rls_obj_type_t objtype

METHOD: lrc_get_lfn
 PARAM: char *pfn
 RESULT: char *lfn
        char *pfn
        ... (may be multiple lfn,pfn tuples)

METHOD: lrc_get_lfn_wc
 PARAM: char *pfn_pattern
 PARAM: globus_rls_pattern_t
 RESULT: char *lfn
        char *pfn
        ... (may be multiple lfn,pfn tuples)

METHOD: lrc_get_pfn
 PARAM: char *lfn
 RESULT: char *lfn
        char *pfn
        ... (may be multiple lfn,pfn tuples)

METHOD: lrc_get_pfn_wc
 PARAM: char *lfn_pattern
 PARAM: globus_rls_pattern_t
 RESULT: char *lfn
        char *pfn
        ... (may be multiple lfn,pfn tuples)

METHOD: lrc_rli_add
 PARAM: char *rli
 PARAM: char *pattern

METHOD: lrc_rli_delete
 PARAM: char *rli
 PARAM: char *pattern

METHOD: lrc_rli_get_part
 PARAM: char *rli
 PARAM: char *pattern
 RESULT: char *rli
        char *pattern
        ... (may be multiple rli,pattern tuples)

METHOD: lrc_rli_info
 PARAM: char *rli
 RESULT: char *url
 RESULT: int updateinterval
 RESULT int flags
 RESULT time_t lastupdate

METHOD: lrc_rli_list
 RESULT: char *rli
        int updateinterval
        int flags

```
        time_t lastupdate
        ... (may be multiple rlis)

METHOD: rli_exists
 PARAM: char *key
 PARAM: globus_rls_obj_type_t objtype

METHOD: rli_get_lrc
 PARAM: char *lfn
 RESULT: char *lfn
        char *lrc
        ... (may be multiple lfn,lrc tuples)

METHOD: rli_get_lrc_wc
 PARAM: char *lfn_pattern
 PARAM: globus_rls_pattern_t
 RESULT: char *lfn
        char *lrc
        ... (may be multiple lfn,lrc tuples)

METHOD: rli_lrc_list
 RESULT: char *lrcurl
        time_t lastupdate
        ... (may be multiple results)

METHOD: set_configuration
 PARAM: char *option
       char *value

METHOD: stats
 RESULT: char *version
        time_t uptime
        int flags
        int lrc_bloomfilterui
        int lrc_lfnlistui
        int lrc_numpfn
        int lrc_nummap
        int rli_numlfn
        int rli_numlrc
        int rli_nummap
```