
# Introduction to writing and debugging micro-services

mwan@diceresearch.org

# Micro-service Input/output parameters


- Prototype of a micro-service

```
int findObjType ( msParam_t *objInput ,
                  msParam_t *typeOutput ,
                  ruleExecInfo_t *rei );
```

- All micro-services only use msParam_t for input/output
- The last input parameter is always ruleExecInfo_t *rei

# Micro-service input/output parameter type - msParam_t


All MS input/output parameters use the same structure

```
typedef struct MsParam {
  char *label;              /* name of the parameter */
  char *type;               /* type of the parameter */
  void *inOutStruct;        /* pointer to value/structure  of
                               the parameter */
  bytesBuf_t *inpOutBuf;      /* optional buffer pointer
                                 for binary values */
} msParam_t;
```

- label
  - Used by the rule engine to identify the parameter
  - Not a concern for MS programming
- type
  - Identifies  the type of data stored in inOutStruct
- inOutStruct
  - pointer to a struct that contains the input/output data
- inpOutBuf
  - Pointer to an optional buffer for large data

# The type field of msParam_t


- The “type” field help to identify the parameters being passed between MSs and client/server.
- Some commonly used types (defined in msParam.h):
- STR_MS_T - string type (most common)
- StrArray_MS_T
- INT_MS_T – integer type
- IntArray_MS_T
- DOUBLE_MS_T
- DataOprInp_MS_T– input struct for data object operation
- CollInp_MS_T
- GenQueryInp_MS_T – input struct for general query
- KeyValPair_MS_T – key/value pair
- GenQueryOut_MS_T
- DataObjInfo_MS_T
- RodsObjStat_MS_T

# msParam helper routines


- Routines to parse and fill in the msParam_t struct
  - Int fillMsParam (msParam_t *msParam, char *label, char *type, void *inOutStruct, bytesBuf_t *inpOutBuf);
    - Generic, fields will only be modified if non-null input. Normally, “label” input is null.
  - Int fillIntInMsParam (msParam_t *msParam, int myInt);
  - Int fillStrInMsParam (msParam_t *msParam, char *myStr);
  - Int resetMsParam (msParam_t *msParam);
    - Free all fields except label.
  - Int parseMspForPosInt (msParam_t *inpParam);
  - char *parseMspForStr (msParam_t *inpParam);
  - Int parseMspForCollInp (msParam_t *inpParam, collInp_t *collInpCache, collInp_t **outCollInp, int writeToCache);
- More helper routines are needed

# Session system parameters


- ruleExecInfo_t *rei
  - A large data structure passed when invoking a rule
  - Implicitly used throughout the rule processing
  - MicroServices can access system parameters in the *rei
  - The structure is defined in reGlobalsExtern.h and it can be extended if necessary
  - Contains various important structures used in the iRODS data management:
    - *rsComm - client-server communication structure
    - *doi - dataObject information
    - *rescGrp - resource (group) informations
    - *uoic - client user information

# Session system parameters


- $ variables – Variables start with “$”
  - Provides a way for rules to reference values in rei structure
  - A mapping from a logical name to values in rei.
  - These mappings are defined in a configuration file:
    - objPath rei->doi->objPath
    - rescName rei->doi->rescName
    - userNameClient rei->uoic->userName
  - These variables can be referenced by rules and MSs
    - assign($rescName, sdsc-samqfs ) /* assign is a microService*/
    - Condition:

      $objPath like /zone/home/sekar@sdsc/nvo/*
    - Parameter passing: findObjType($objName,*Type)

# Writing Micro-services


- Typically MS codes are short
- Call existing server routines
  - Reasonably familiar with server routines
  - Server API handler routines
    - A large number of APIs for clients to request services from servers
    - Prototype of APIs and API handlers are given in the lib/api/include directory
    - Each client API has one server API handler
      - In dataObjOpen.h : rcDataObjOpen() and rsDataObjOpen()
      - To open an iRods file on the server, call rsDataObjOpen

# A micro-service example (msiCollRepl in reDataObjOpr.c


```
int
msiCollRepl (msParam_t *collection, msParam_t *targetResc, msParam_t *status,
      ruleExecInfo_t *rei)
{
        /* for parsing collection input param */
        collInp_t collInpCache, *collInp;

        /* to pass as parameter to rsCollRepl */
        dataObjInp_t collReplInp;

        /* misc. to avoid repeating rei->rsComm */
        rsComm_t *rsComm;
        /*************************************  INIT ************************************/

        /* For testing mode when used with irule --test */
        RE_TEST_MACRO ("    Calling msiCollRepl")

        /* Sanity checks */
        if (rei == NULL || rei->rsComm == NULL) {
                rodsLog (LOG_ERROR, "msiCollRepl: input rei or rsComm is NULL.");
                return (SYS_INTERNAL_NULL_INPUT_ERR);
        }

        rsComm = rei->rsComm;
b
```

# A micro-service example (msiCollRepl in reDataObjOpr.c) -cont


```
/******************************** PARAM PARSING  *************************/

    /* Parse target collection */
    rei->status = parseMspForCollInp (collection, &collInpCache, &collInp, 0);

    if (rei->status < 0) {
            rodsLog  (LOG_ERROR,
              "msiCollRepl: input collection error. status = %d", rei->status);
            return (rei->status);
    }


    /* Parse resource name and directly write to collReplInp */
    rei->status = parseMspForCondInp (
       targetResc, &(collReplInp.condInput),         DEST_RESC_NAME_KW);

    if (rei->status < 0) {
    rodsLogAndErrorMsg (LOG_ERROR, &rsComm->rError, rei->status,
            "msiCollRepl: input inpParam2 error. status = %d", rei->status);
        return (rei->status);
}
```

# A micro-service example (msiCollRepl in reDataObjOpr.c) -cont


```
/************************* SERVER API CALL**********************/

    /* Copy collection path to input struct */
    strncpy (collReplInp.objPath, collInp->collName, MAX_NAME_LEN);

    /* Call rsCollRepl() */
    rei->status = rsCollRepl (rsComm, &collReplInp, NULL);



    /************************* OUTPUT PACKAGING **************/

    /* Send out op status */
    fillIntInMsParam (status, rei->status);


    return (rei->status);
}
```

# Writing micro-services – putting it all together


- Adding a MS routine msiCollRepl to an existing file reDataObjOpr.c
- Add the prototype of msiCollRepl to reDataObjOpr.h
  - Int msiCollRepl (msParam_t *collection, msParam_t *targetResc, msParam_t *status, ruleExecInfo_t *rei);
  - Add a line to the MicrosTable[] in reAction.h

…...

……

{"msiRmColl",3,(funcPtr) msiRmColl},

{"msiReplColl",4,(funcPtr) msiReplColl},

{"msiCollRepl",3,(funcPtr) msiCollRepl},

# Adding a new Micro-service module


- Modules are a set of optional MSs that can be compiled and linked with the server
- https://www.irods.org/index.php/How_to_create_a_new_module
- The “modules” directory contains all the optional MS modules
  - hdf5, images, ERA
- Create a new directory for your module
  - Easiest just to copy the entire directory of an existing module for the structure
- Edit the Makefile to include the your MS directories and object files
- Build the server with your module, do either:
  - ./configure --enable-myModule
  - Edit the config/config.mk file by add an entry in the MODULES definition
    - MODULES= properties hdf5 myModule

# Debugging a Micro-service routine


- **Server debugging**
- **Printf type debugging**
  - rodsLog() function – print to the log file
- **Gdb**
  - Run an example of stepping through the msiCollRepl() routine on the server
  - On client

    **gdb irule**

    (gdb) **break clientLogin**

    Breakpoint 1 at 0x804adb2: file /data/mwan/rods/iRODS/lib/core/src/clientLogin.c, line 97.

    (gdb) **run -F collRepl.ir**

    Breakpoint 1, clientLogin (Conn=0x8ee6508)

    at /data/mwan/rods/iRODS/lib/core/src/clientLogin.c:97

    97 if (Conn->loggedIn == 1) {

    (gdb)
  - At this point, the client irule process is stopped and an irodsAgent process has been created.

# Debugging a Micro-service routine (cont)


- On the server machine:

srbbrick8-4% **ps -elf | grep irodsAgent**

0 S mwan 1435 24013 0 76 0 - 1587 schedu 10:24 ? 00:00:00 irodsAgent
0 S mwan 9883 24013 0 81 0 - 1578 schedu 16:32 ? 00:00:00 irodsAgent
0 S mwan 10779 10737 0 75 0 - 405 pipe_w 16:40 pts/7 00:00:00 grep irodsAgent

- **Pick the latest irodsAgent process – 9883**

srbbrick8-4% **gdb irodsAgent 9883**
(gdb) **break msiCollRepl**
Breakpoint 1 at 0x80b6b46: file /data/mwan/rods/iRODS/server/re/src/reDataObjOpr.c, line 1947.
(gdb) **cont**

- **The server now is waiting for the client request**

- **Go back to the client: Type in “cont” to continue**

(gdb) **cont**
Continuing.

# Debugging a Micro-service routine (cont)


- **Go to the server gdb session and it should breakpoint in misCollRepl():**

```
Breakpoint 1, msiCollRepl (collection=0x9300bd8, targetResc=0x9300fe0,
    status=0x9301458, rei=0xbfff63e0)
    at /data/mwan/rods/iRODS/server/re/src/reDataObjOpr.c:1947
1947            RE_TEST_MACRO ("    Calling msiCollRepl")
(gdb)
```

- **Type in “next” to step through the code, “list” to list the code, etc**

(gdb) **next**

```
1971            rei->status = parseMspForCondInp (targetResc, &(collReplInp.condInput),
      DEST_RESC_NAME_KW);
(gdb)
1973            if (rei->status < 0) {
(gdb)
1984            strncpy (collReplInp.objPath, collInp->collName, MAX_NAME_LEN);
(gdb)
1987            rei->status = rsCollRepl (rsComm, &collReplInp, NULL);
```

- **Use the “print” to examine the values of variable**

(gdb) **print collReplInp**

```
$1 = {objPath = "/tempZone/home/rods/testdir", '\0' <repeats 1060 times>,
  createMode = 0, openFlags = 0, offset = 0, dataSize = 0, numThreads = 0,
  oprType = 0, specColl = 0x0, condInput = {len = 1, keyWord = 0x9300828,
    value = 0x9300858}}
```

(gdb) **print *collection**

```
$2 = {label = 0x9301470 "*C", type = 0x9301480 "STR_PI",
  inOutStruct = 0x93014a0, inpOutBuf = 0x0}
```