

GT 4.2.0 Migrating Guide for XIO

Table of Contents

1. Migrating XIO from GT4.0.x	1
2. Migrating XIO from GT3	2
3. Migrating XIO from GT2	2

<titleabbrev>Migrating Guide</titleabbrev>

The following provides information about migrating from previous versions of the Globus Toolkit.

1. Migrating XIO from GT4.0.x

The only significant change to Globus XIO in this release is the addition of "string attributes". The previous method of handling attributes still works in an entirely backward compatible way, but the string attributes adds a much needed convenience. Drivers can now provide a list of key=value pairs. A user can then specify these options at the command line and at run time. This make run time configuration much more flexible.

A driver can choose to expose parameters in a string form. Providing this feature makes dynamically setting driver specific options much easier. A user can then load the driver by name and set specific options by name all at runtime with no object module references. For example, a TCP driver can be loaded with the string *tcp* and the options can be set with the string *port=50668;keepalive=yes;nodelay=N*.

This would set the port to 50668, keepalive to true and nodelay to false. The particular string definition is defined by the tcp driver. This is done by creating a `globus_i_xio_attr_parse_table_t` array within the drivers source code. See the TCP driver source code for an example. Each row of the array is 1 option. There are 3 sub-members of each row entry: key, cmd, and parse function. The key is a string that defines what option is to be set. In the above example string *port* would be a key. *cmd* tells the driver what cntl is associated with the key. In other words, once the string is parsed out it converts the key into a driver specific control enum. XIO then uses that enum as it would if a driver specific option was set in the source code. For more information on controls see `globus_xio_attr_cntl`.

The final value in the array entry is the parsing function. The passing function takes the value of the key=value string and turns it into the specific data type associated with that function. There are many available parsing functions but the developer is free to right their own. Here are some examples:

- `globus_i_xio_attr_string_single_bool`
- `globus_i_xio_attr_string_single_float`
- `globus_i_xio_attr_string_single_int`
- `globus_i_xio_attr_string_single_string`
- `globus_i_xio_attr_string_dual_positive_int`

4.2.0 No other differences will be noticed by those previously familiar with `globus_xio`.

2. Migrating XIO from GT3

Globus XIO has made very few changes in the API since its introduction. Users of GT 4.2.0 who are previously familiar with `globus_xio` will notice no difference.

3. Migrating XIO from GT2

Globus XIO is a new component as of Globus 3.2. Previous versions of Globus provided an API called `globus_io`. XIO is a replacement for `globus_io`; however, a compatibility layer exists to allow full backward compatibility for applications written against `globus_io`. For users familiar with `globus_io` the learning curve for XIO will be very small. The APIs use the same event model and only significantly differ in connection establishment.

DRAFT