DCAF Engine-to-Engine (E2E) Tag Exchange Module

The DCAF Engine-to-Engine (E2E) Tag Exchange Module is a plug-in module for the Distributed Control and Automation Framework (DCAF). Using this module requires a basic understanding of the principles of DCAF, including but not limited to: installing DCAF plugins, modifying a DCAF configuration using the Standard Configuration Editor, and running a DCAF configuration using the Engine Execution Interface. Getting Started Guides and general information about DCAF can be found on the DCAF Community Page.

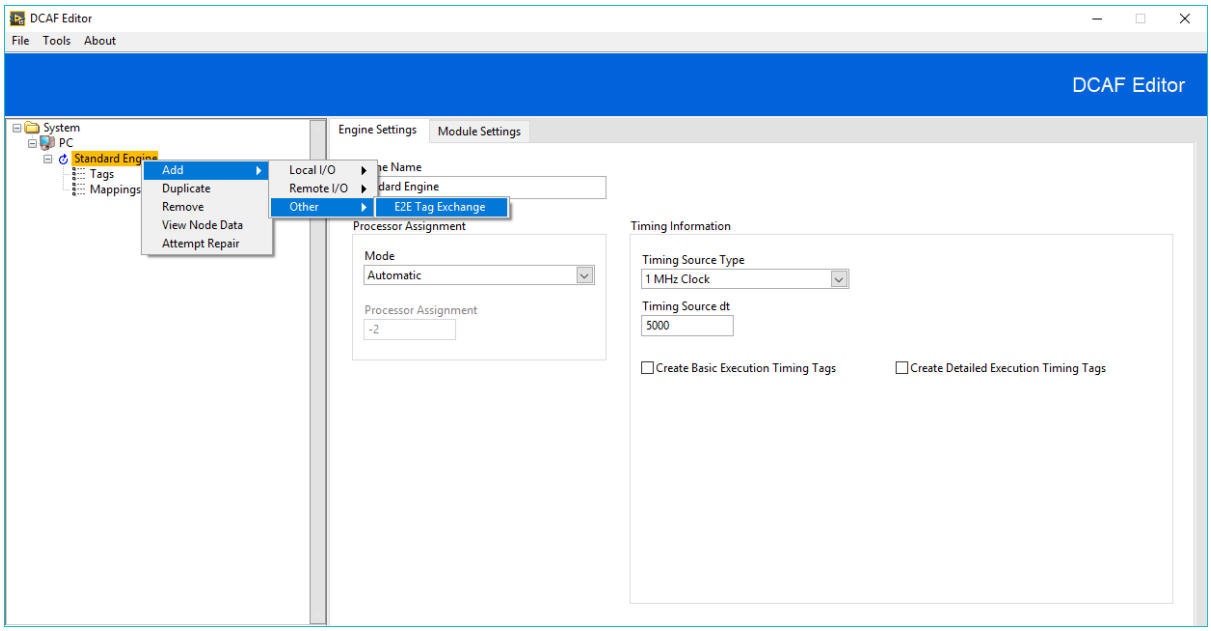
Overview

The Engine-to-Engine Tag Exchange (E2E) module is meant to simplify the exchange of tags between separate DCAF engines. Rather than configuring output and input channels manually using a specific I/O Module to exchange tags, the E2E module handles channel configuration automatically when a tag is selected to be shared between engines. The editor node of the E2E module also allows the user to determine the transfer mechanism for tag exchange. The transfer mechanism determines the allowable data types, the pairing depth (inner-target, inter-target, both), and other specific configuration information. The module was designed to allow for users to create and add new transfer mechanisms. The first two implemented mechanisms were UDP and Single Element RT FIFOs.

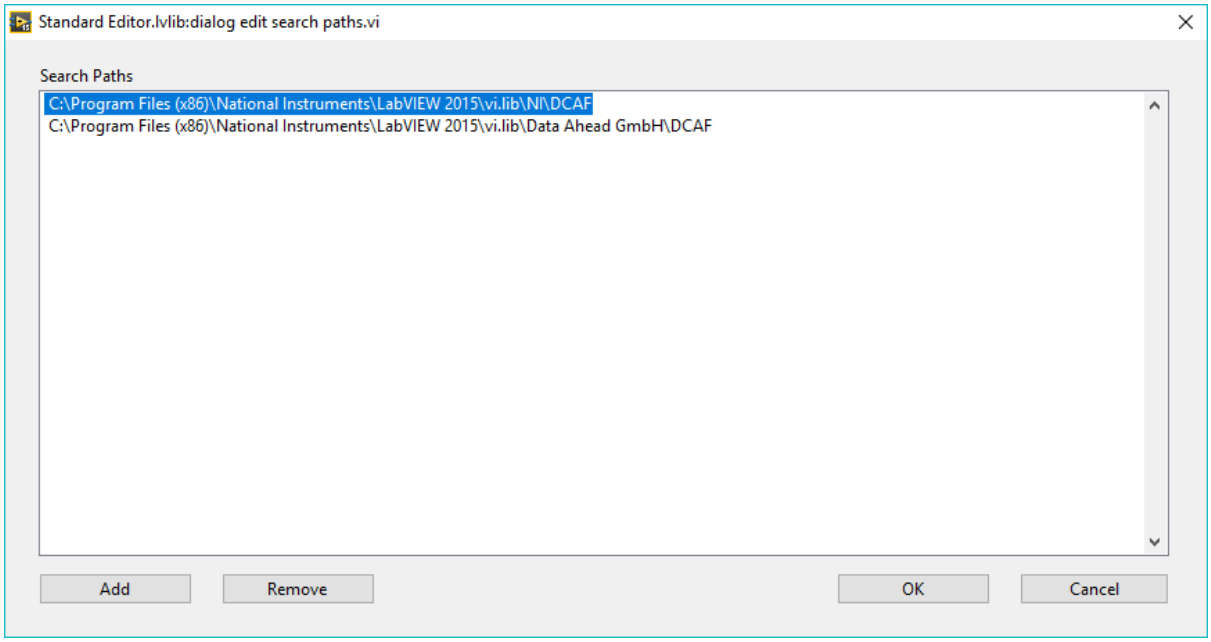
Using the Module in the Standard Configuration Editor

Adding the Module to a DCAF Configuration

The VI Package Manger installer for the E2E Module will attempt to add the E2E Module's destination directory to the DCAF Configuration Editor's plugin search paths. If successful, you will be able to add the E2E Module to your DCAF Engine's configuration by right clicking the engine configuration and selecting Add>>Other>>E2E Tag Exchange.

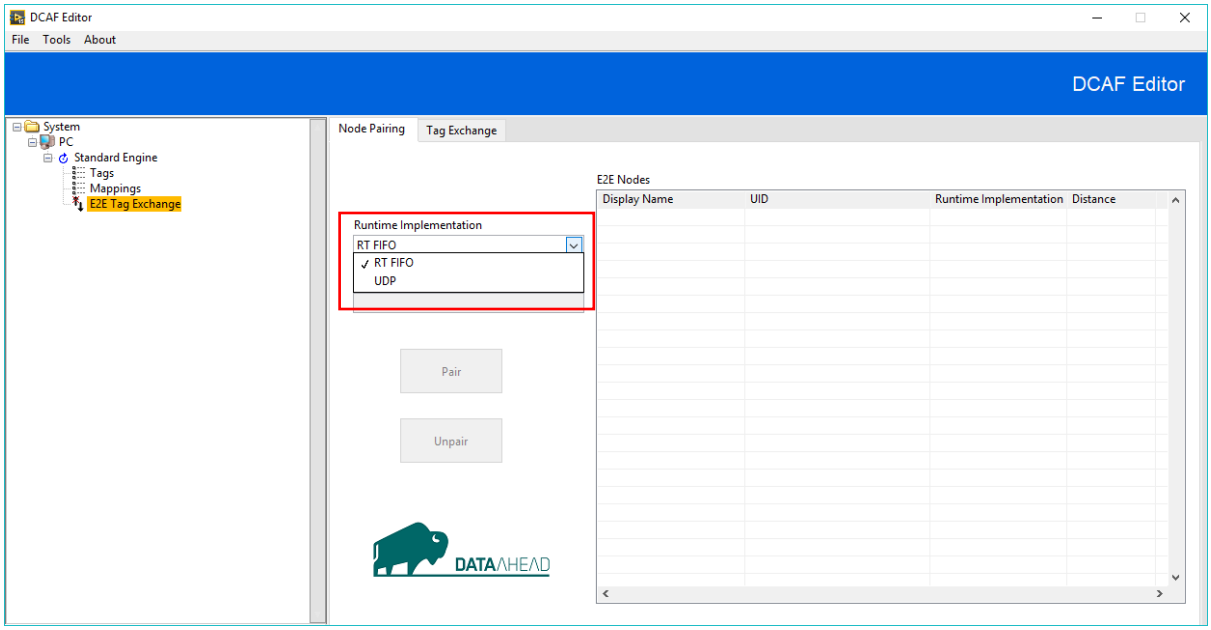


If not, you will need to add the E2E Module's destination directory to the Plugin search paths manually. The default installation directory for the E2E module is *\<LabVIEW>\vi.lib\Data Ahead GmbH\DCAF\E2E Tag Exchange Module*. Add the directory to the plugin search paths by selection Tools>>Edit Plugin Search Paths... from the toolbar of the DCAF Standard Configuration editor.



Selecting a Runtime Implementation

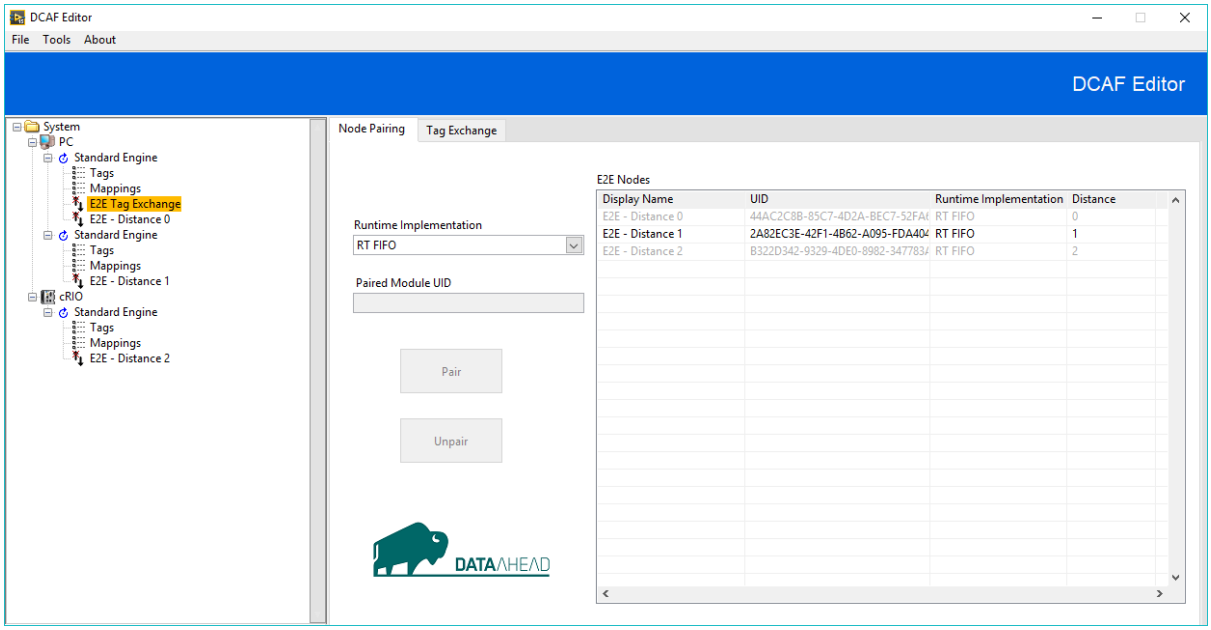
Once you have added the E2E Module to a DCAF Engine, you can select its runtime implementation by using the drop-down box on the Node Pairing tab of the configuration window.



The runtime implementation determines which runtime component is used to exchange the tags, which pairing distances are acceptable, the data types that can be exchanged, and other specific configuration information.

Pairing the Module

E2E Modules exchange tags by pairing with each other on a One-to-One basis. When an E2E module is selected, the “E2E Nodes” table will display all other E2E modules in the configuration.



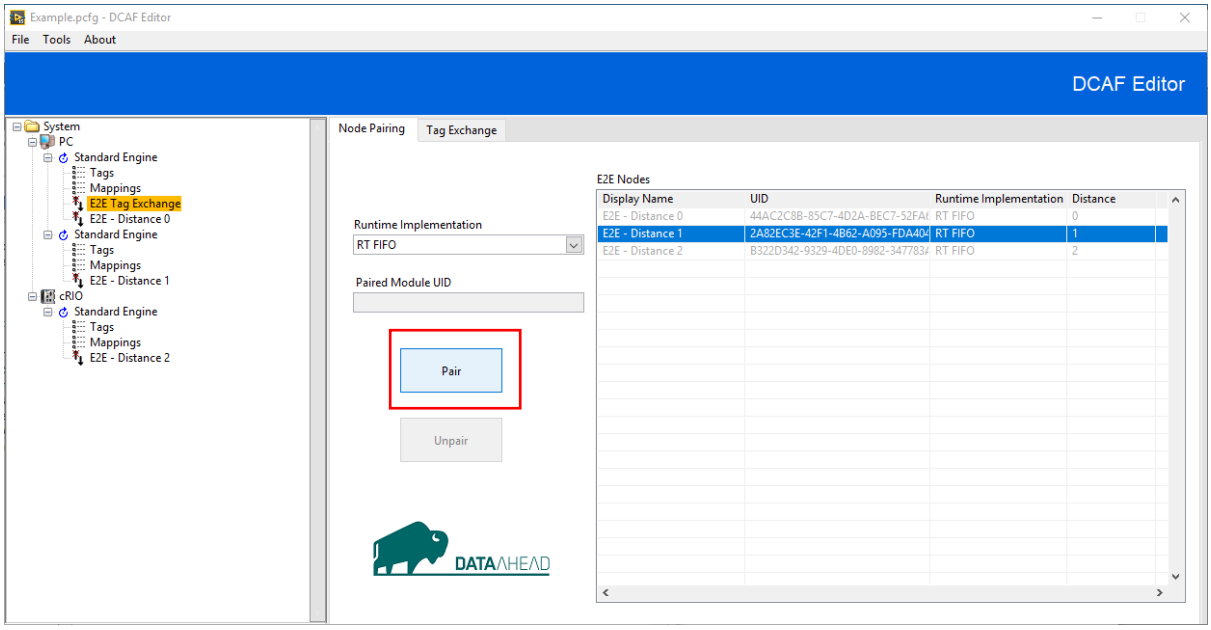
The information displayed for each module is:

1. Display Name – The module’s current alias
2. UID – The unique identifier of the module
3. Runtime Implementation
4. Distance – The relationship of the selected E2E module to the module in the table
    a. 0 – The module is under the same engine
    b. 1 – The module is under a different engine in the same target
    c. 2 – The module is under a different engine in a different target

The row for a module will be disabled given one of the following conditions:

1. The runtime implementation is different
2. The module is already paired
3. The runtime implementation doesn’t support the pairing distance (ex: RT FIFOs can’t exchange tags between separate targets)

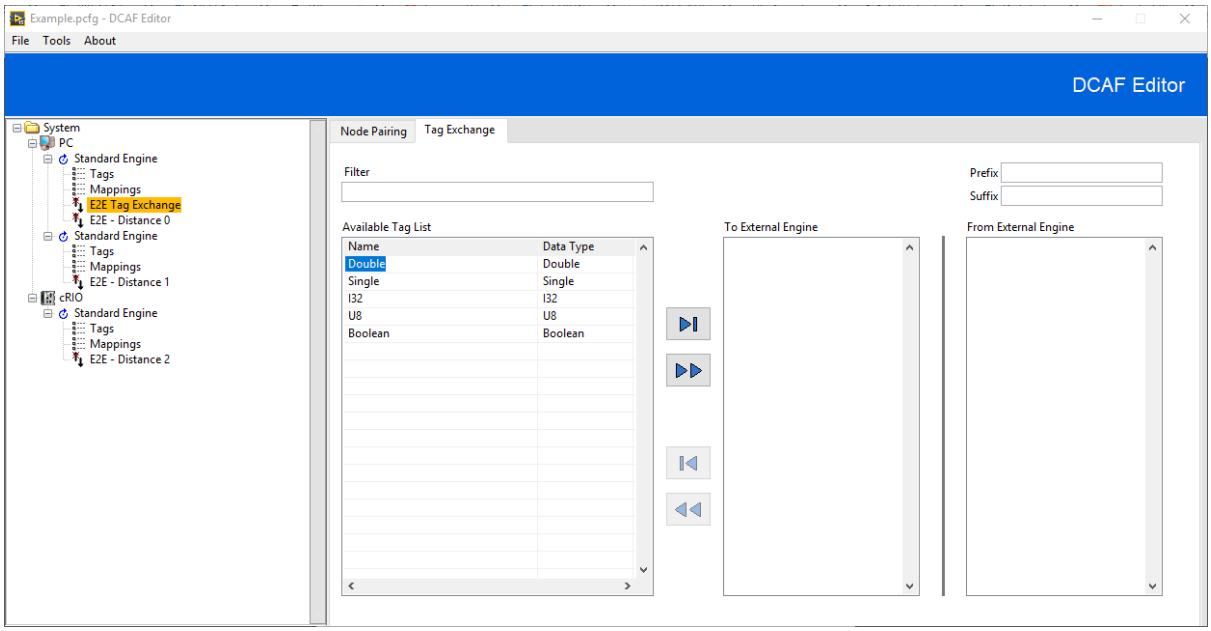
Pair a module by selecting an enabled row in the table and clicking “Pair”. Use the “Unpair” button to unpair the paired modules.



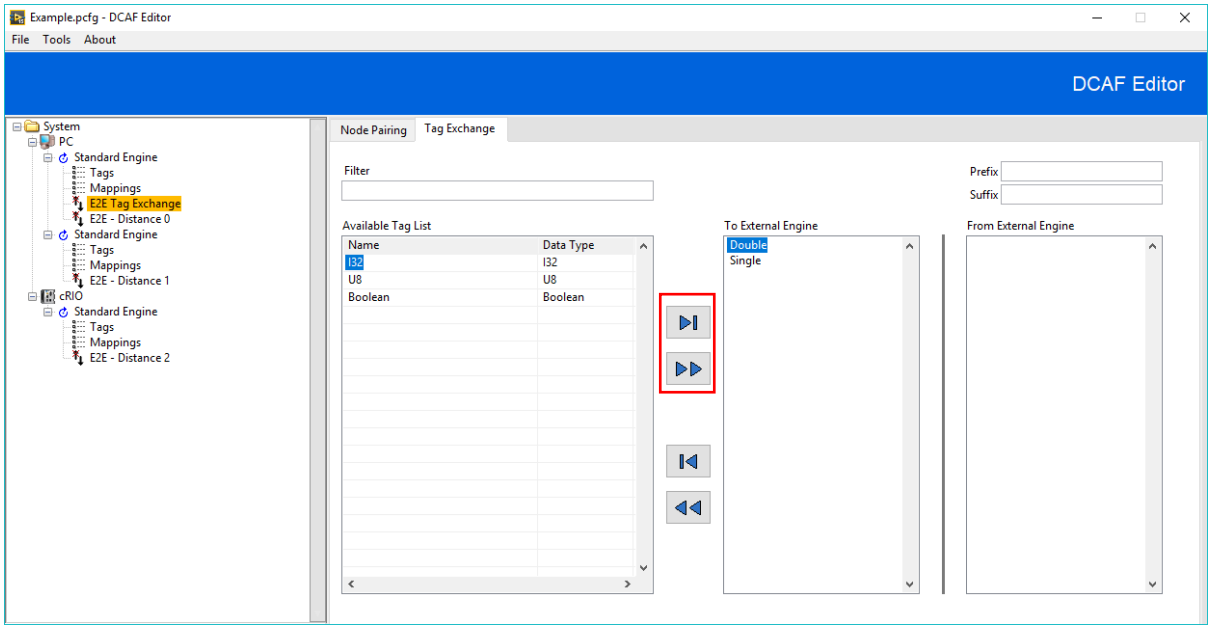
Exchanging Tags

The “Tag Exchange” tab displays the following information:

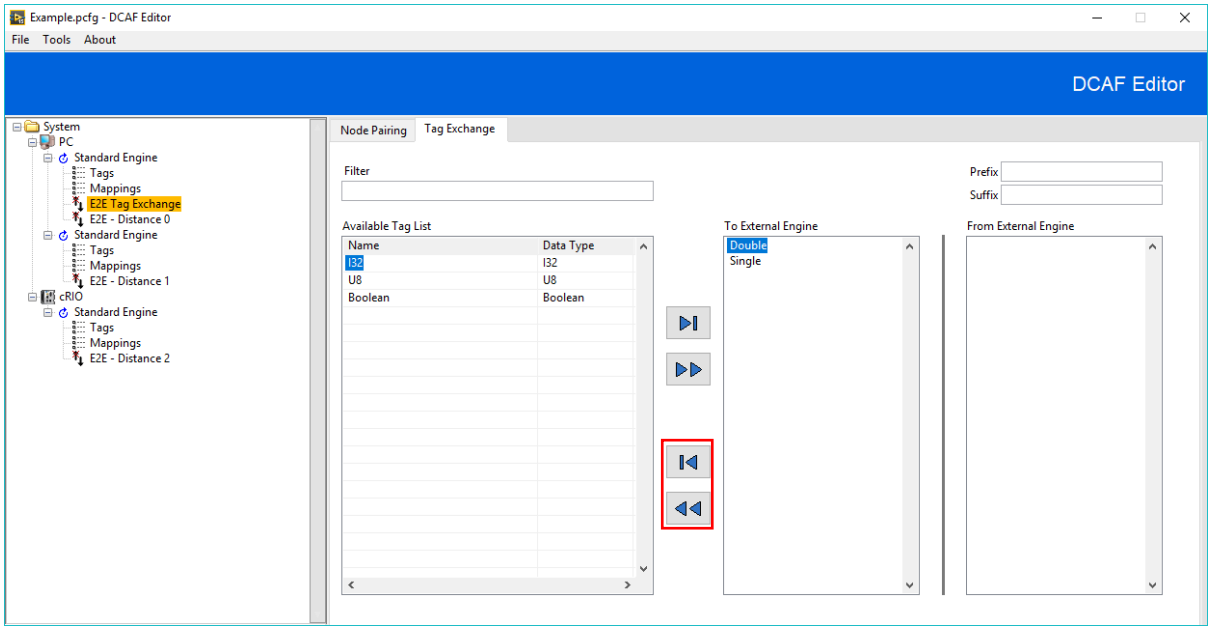
1. Available Tag List – The tags that are currently available for exchange
2. To External Engine – Tags currently configured for transfer to the external engine
3. From External Engine – Tags coming from the external engine
4. Filter – A string input used to filter results with a match pattern function
5. Prefix – A string concatenated to the front of all input tag names
6. Suffix – A string concatenated to the back of all input tag names



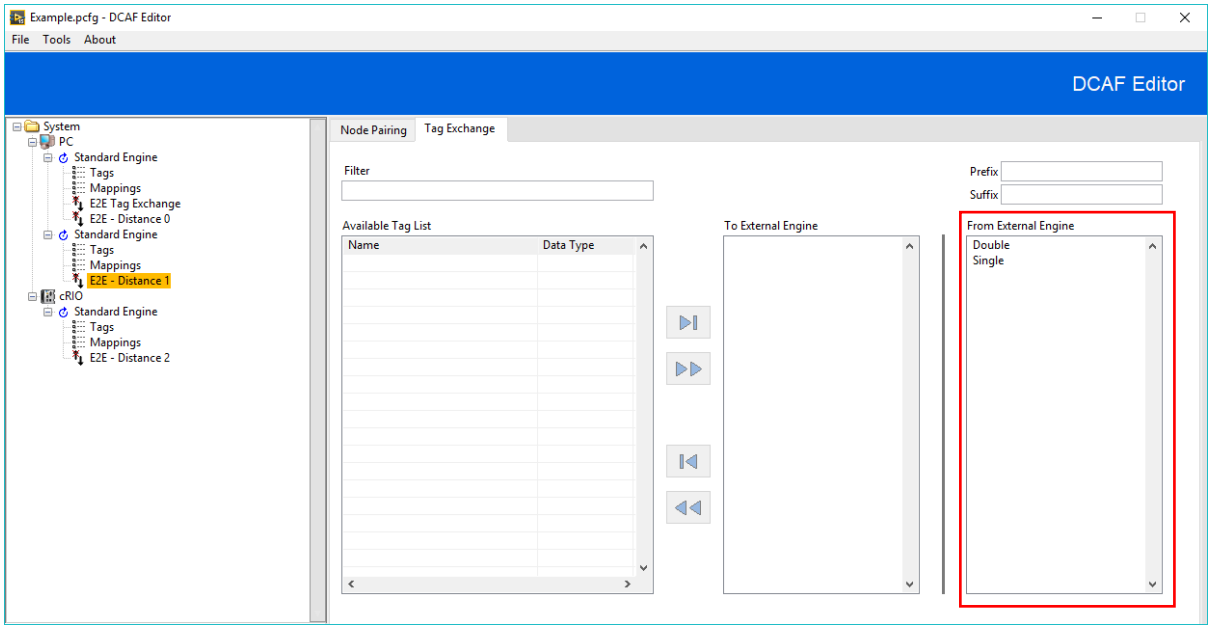
Tags are considered available when the data type is supported by the runtime configuration and the tag isn’t already being transferred either to or from the current module. To transfer a tag, select the tag from the available tag list and click the “Move one right” button. Transfer all tags by clicking the “Move all right” button.



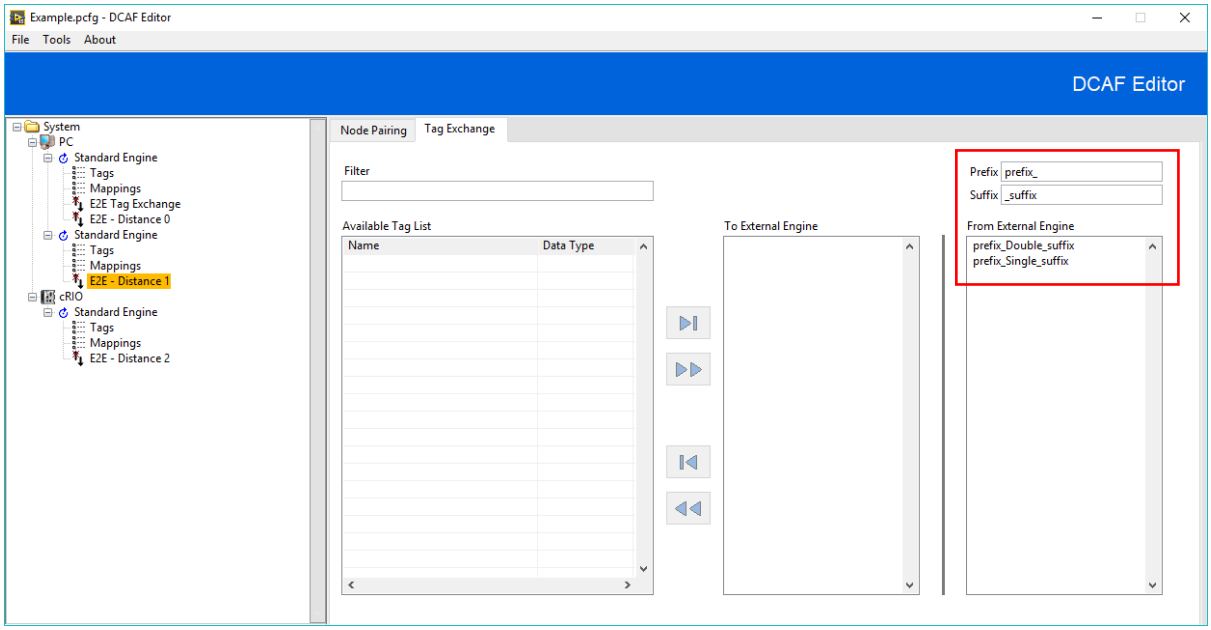
Remove tags from the list by selecting a tag from the “To External Engine” List and using the “Move one left” and “Move all left” buttons.



The tags will now appear in the “From External Engine” list in the paired module.

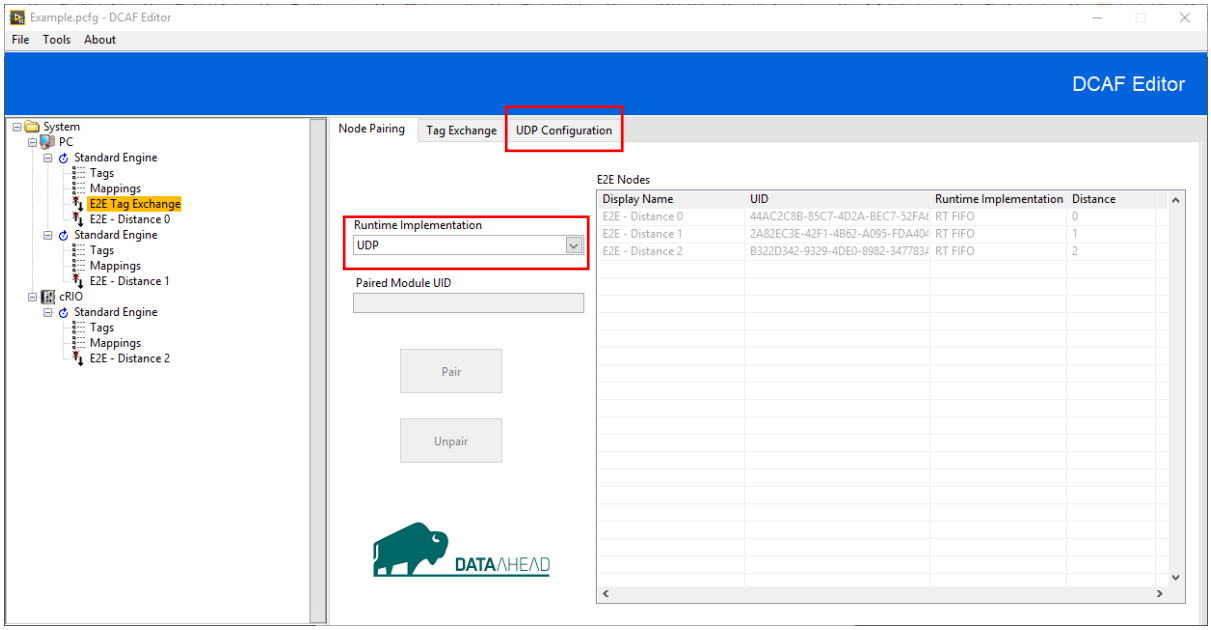


Renaming tags corresponding to each input/output channel is not supported because the E2E module uses each tag's name to sort tags for transfer. Use the "Prefix" and "Suffix" inputs to manage tags coming from an external engine.

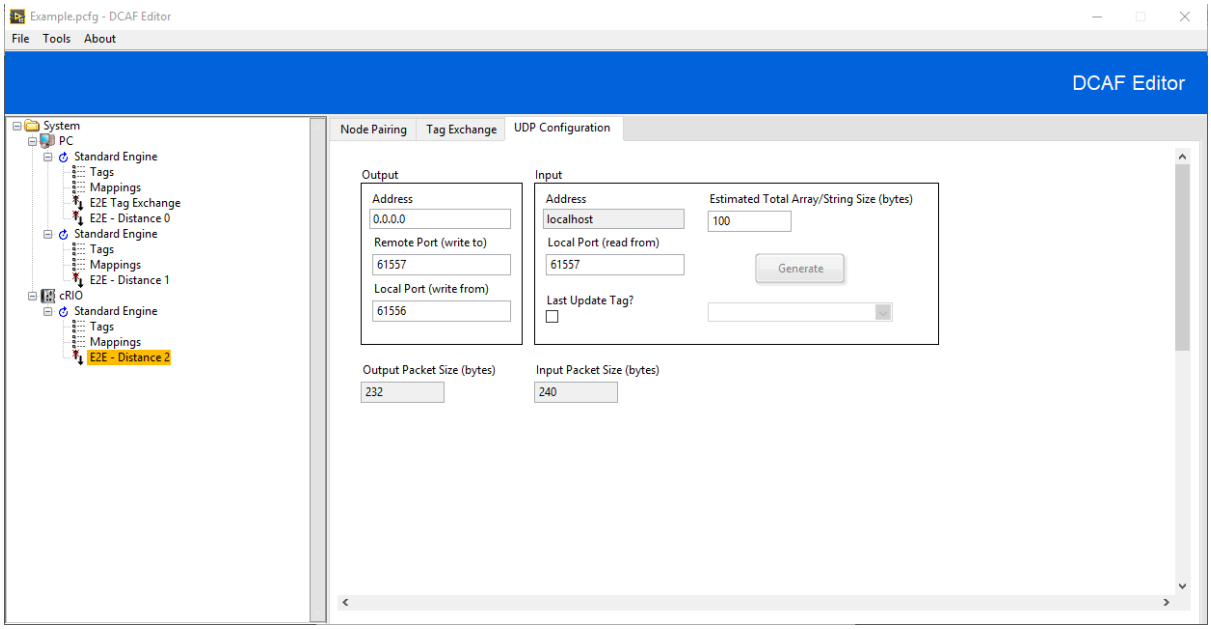


Editing Transfer Mechanism Specific Configuration

If a transfer mechanism has specific configuration information, a new tab will be available.



UDP Configuration

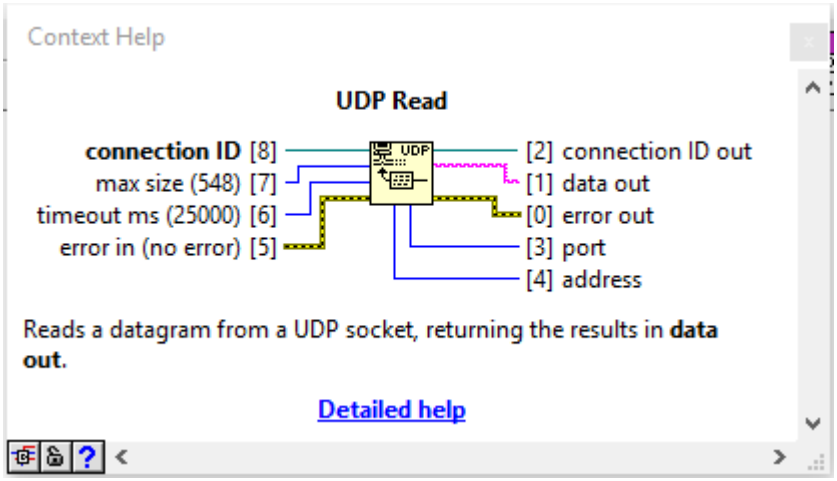


The UDP runtime implementation has the following specific configuration information:

1. Output Configuration
    a. Address – The network address to write to
    b. Remote Port (write to) - The UDP port to write to
    c. Local Port (write from) – The UDP port to write from
2. Input Configuration
    a. Address – Always localhost
    b. Local Port (read from) – The UDP port to read from
    c. Last Update Tag? – If selected a channel of type double will be created. Use the corresponding drop-down selector to map a tag to the channel. The channel

contains a timestamp with the last successful read time. The generate button will create and map a tag to the channel

d. Estimated Total Array/String Size (bytes) – This input will be available if the tags from the external engine contain data types of dynamic sizes (Strings and Arrays). The UDP runtime component will attempt to read a packet from the input port with a max size determined by the size of the statically sized tags, the packet header, and the value contained in this input plus a small buffer



3. Estimated Output Packet Size (bytes) – The combined size, in bytes, of all output tags with statically sized data types plus the output header
4. Estimated Input Packet Size (bytes) – The combined size, in bytes, of all input tags with statically sized data types, plus the input header, plus the value in the Estimated Total Array/String Size (bytes) field

Modifying the Plug-in

Editing the Source

When the E2E module is installed, all compiled files are place in vi.lib for your selected version of LabVIEW (<Program Files>\<LabVIEW>\vi.lib\Data Ahead GmbH\DCAF\E2E Tag Exchange Module). These installed files should not be modified in vi.lib without consideration. The recommended method for editing the module is as follows:

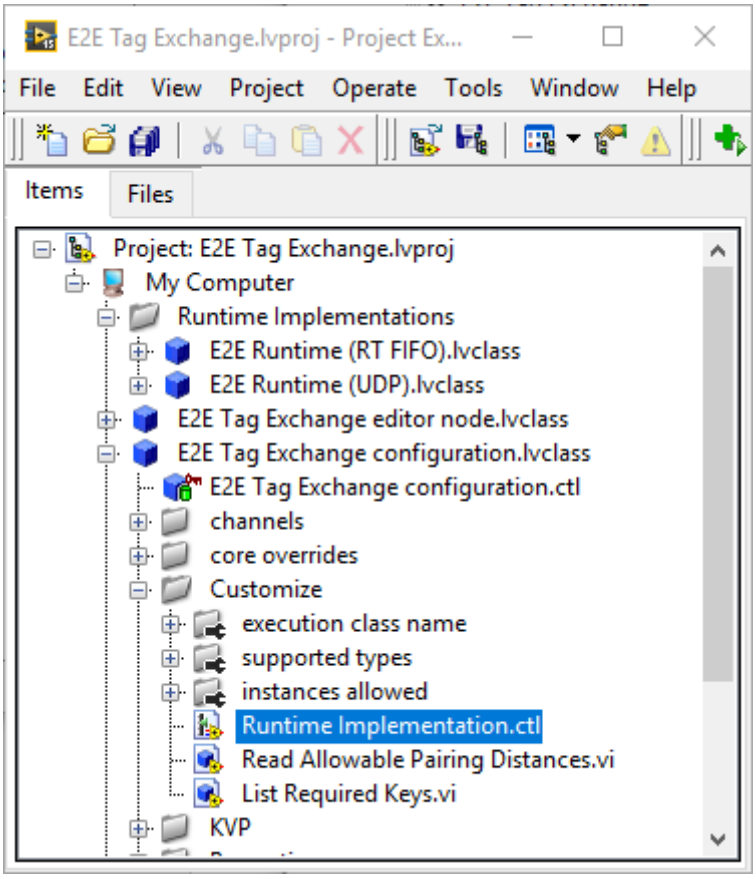
1. Uninstall the compiled E2E Module using VI Package Manager
2. Fork or Clone the source from GitHub: https://github.com/LabVIEW-DCAF/E2E
3. Modify the source
4. (Optional) Add the intended modifications to the issue tracker and create a Pull Request on GitHub if you want to contribute the changes back to the community
5. Either add the source directory to the Standard Configuration Editor search paths or build and install a new VI Package using the VI Package Build Specification contained in the source

Adding Additional Transfer Mechanisms

The E2E Tag Exchange Module was designed to allow for additional transfer mechanisms to be implemented. The E2E module uses case structures wired to the “Runtime Implementation.ctl” type definition without a “Default” case to force the developer to override necessary functionality. After overriding all necessary behavior for the configuration and editor components of the E2E module, the runtime component of the new transfer mechanism needs to be developed. VIs requiring modification will typically be found in virtual folders named “Customize” in the LabVIEW project explorer.

1. Modify E2E Tag Exchange configuration.lvclass

Begin by adding the name of the new transfer mechanism to the “Runtime Implementation.ctl” type definition.



After applying changes to the type definition, the following configuration class methods need to be edited:

get supported types.vi

Provides an array of the data types that is supported by the current runtime implementation.

instance count allowed.vi

Provides the number of module instances allowed per engine based on the current runtime implementation. -1 corresponds to unlimited instances.

Read Allowable Pairing Distances.vi

Provides an array of the pairing distances that are supported by the current runtime implementation. Acceptable values for each array element are 0, 1, and 2. 0 is an E2E module in the same engine, 1 is an E2E module in a different engine in the same target, and 2 is an E2E module in a different engine in a different target.

get channels.vi

Returns an array of channels currently configured by the module. If your runtime implementation supports additional channels besides the tag exchange channels, add them here. See the UDP implementation for an example. Do not add any additional input or output channels as the channel sorting and tag exchange between modules may be incorrect.

List Required Keys.vi

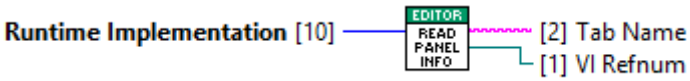
The E2E module uses key-value pairs to store runtime implementation specific configuration. If your runtime implementation has additional configuration information, add the keys and initial values here. Use the “Read Key.vi” and “Write Key.vi” methods in the E2E configuration class to access the key-value pairs.

execution class name.vi

Returns the class name of the runtime component for the current runtime implementation. Return the name of the class you will create in the following section.

2. Modify E2E Tag Exchange editor node.lvclass

The following editor class method needs to be edited:



Read Panel Info.vi

If your runtime implementation has specific configuration information, return the name you would like for the configuration tab and a reference to a VI that is used to modify the configuration information. A template VI has been created to define a VI server interface and to simplify the creation of the VI. Create a new VI from the following template VI included in the source:

See the UDP runtime component for an example.

3. Create the Runtime Class

Create a new LabVIEW class that inherits from TBM module execution interface.lvclass. The input and output channels configured by the editor and loaded during the init.vi override will match properly between paired modules. Use Read Key.vi in the init.vi override to access specific runtime configuration. No other logic is necessary for the E2E tag exchange. See the UDP runtime class for an example.