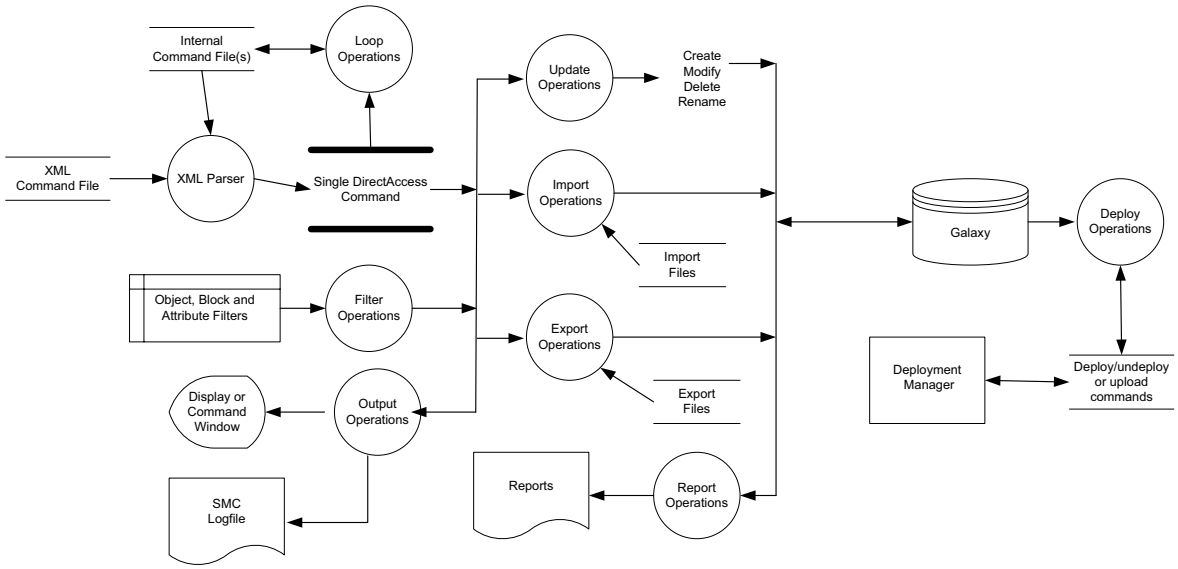
Foxboro Evo™ Process Automation System

Product Specifications

Foxboro by Schneider Electric

PSS 31S-10SCRIPT

Scripting with Direct Access



Control configurations can be manually updated but widespread updates can be time consuming and error-prone. The Direct Access scripting tool permits bulk creation, update, and deployment of a revised configuration. This reduces engineering effort and improves the consistency of the resulting database.

OVERVIEW

The Direct Access Query/Scripting application enables the user to programmatically create, rename, deploy, undeploy import, export and delete derived template and instance representations of Controllers, Devices, FBMs, FCMs, Workstations, Network Switches, Blocks and Compounds as well as Strategies, Strategy Declarations, PlantUnit nodes, EquipUnit nodes, Application Objects & User Attributes and Software Packages.

The user can also update their attributes [parameters] and report on their current settings. The user can also construct hierarchical representations through assignment. For example, assign a Block to a Strategy, assign a Strategy to a Compound, assign a Compound to a Controller, assign an FBM to a Controller.

FEATURES

The Direct Access Query/Scripting tool enables the user to:

- Create new derived objects, instances, or templates. Examples include: Blocks, Strategies, Compounds, Stations, Modules, Devices.
- Modify the attributes (parameters) of Foxboro Evo™ elements such as Block or Compound parameters.
- Delete existing derived objects.
- Construct a Foxboro Evo system database configuration by assigning components to containing elements, such as the assignment of:
  - Blocks to Strategies.
  - Strategies to Compounds.
  - Compounds to Controllers.
  - Fieldbus Modules to Controllers.
- Report upon the configuration of all or a portion of a Foxboro Evo system database such as all of the Blocks in a Strategy or all of the Strategies in a Compound.
- Apply multiple criteria (Filters) to queries to limit the returned information to what is required.
- Combine multiple queries in a single script.
- Dynamically compute the name of an Object in a script using regular expressions.
- Repeat the execution of a set of operations in a script.
- Manage the execution time of operations within a script.

BENEFITS

- Reduce the elapsed time it takes to complete a set of repetitive tasks because the tasks are performed at computer speed instead of performing manual operations
- Reduce the engineering hours it takes to complete a set of complex repetitive tasks
- Reduce the errors typically incurred by performing highly repetitive manual operations
- Generate a complex set of reports from a single script that can be stored
- Perform specified operations, such as back up the Galaxy, periodically or based on the occurrence of an event such as the time is 2 AM.

USER INTERFACE

The user can either execute a Direct Access query directly through the command line interface as shown below:

DirectAccess_Cmd <GalaxyNode> <GalaxyName> <CommandFile> <UserName> <Password>

Or execute a script using the Direct Access dialog box as shown in Figure 1.

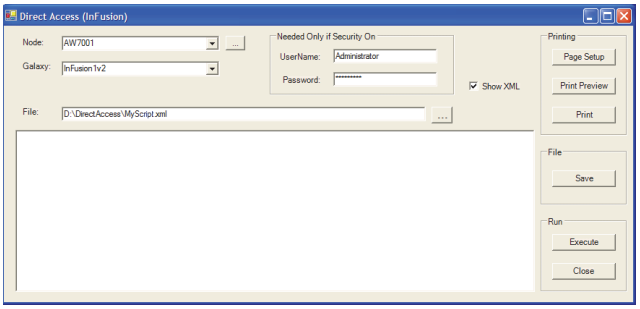


Figure 1. Direct Access Dialog Box

In both cases the Command File is stored as an XML document. An example that creates a new Strategy Template called $Cascade that is derived from the base template called $Strategy is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<DirectAccess>
        <CreateStrategy Template="$Strategy"
         Strategy="$Cascade"/>
</DirectAccess>
```

Direct Access provides a rich set of tools to enable these command files to perform large scale bulk edits on the Galaxy Repository database. These include the ability to apply multiple Filters against a query, the ability to perform repetitive operations and the ability to apply string and or value substitutions against a named Object or Attribute.

QUERY FILTERS

A query filter enables the user to specify criteria when searching for software objects and/or attributes. You can specify filters with standard expressions.

With Direct Access querying capabilities, you can use the results of a filter in another operation such as generating a report, returning a value, updating an attribute, or performing an action on an object or attribute.

The three types of query filters for Direct Access are:

- QueryFilter - search for objects such as a Controller, Device or Strategy instance or derived template
- BlockQueryFilter -define which block and ECB instances to return
- AttributeQueryFilter -define which attributes, e.g. Block parameter to return.

Query filters are constructed using Filter Conditions. These describe the search criteria to be applied by the filter. Direct Access utilizes five filter conditions. These are:

- BasedOn -All blocks/ECBs satisfying this filter are based on the base template specified by this base template.
- ContainedBy - All blocks satisfying this filter are contained by the strategy specified by this tagname. NOTE: The ContainedBy condition can only be used for the WhereUsed report.
- DerivedOrInstantiatedFrom - All blocks/ECBs satisfying this filter are derived or instantiated from the template specified by this tagname.
- NamedLike - All blocks/ECBs satisfying this filter have a tagname that satisfies the specified LIKE clause.
- NameEquals - All block/ECBs satisfying this filter have a tagname exactly matching the specified value.
- Hierarchical NamedLike - all blocks/ECBs satisfying this filter have a contained name that satisfies the specified LIKE clause.

Two of the Filter Conditions listed above introduce the keyword NamedLike. This clause provides the ability to specify a set of Object that satisfy a regular expression based on one of the following special characters. These are:

- % - any string of zero or more characters, e.g., NamedLike="AW%" will return all objects with tagnames that begin with the characters "AW"; NamedLike="%AW%" will return all object instances that have the characters "AW" anywhere in the tagname.
- _ (underscore) - any single character, e.g., NamedLike="AW000_" will return all objects with tagnames that begin with the characters "AW000" followed by any other single character.

- [ ] - any single character in the specified range, e.g., NamedLike="[ACS]%" will return all objects with tagnames that begin with the characters "A", "C" or "S".
- [^] - any single character NOT in the specified range, e.g., NamedLike="[^ACS]%" will return all objects with tagnames that do not begin with the characters "A", "C", or "S".

Query filters can be combined to accomplish multi-criteria searches. Multiple QueryFilter commands can be issued against the same FilterName, resulting in a filter with an implied AND between the multiple dissimilar conditions, and an implied OR between multiple identical conditions.

This is best explained by example. Consider a query returning all objects that are:

- Based on $FCP270
- OR Based on $AW70P
- OR Based on $ZCP270
- AND Named like XYZ% (that is, letterbugs beginning with the characters XYZ)
- AND NOT assigned to equipment unit Equip_Unit_001.

The filter statements in the Command File would look like this:

```
<QueryFilter Filter="Filter1" Condition="BasedOn"
Value="$FCP270"/>
<QueryFilter Filter="Filter1" Condition="BasedOn"
Value="$AW70P"/>
<QueryFilter Filter="Filter1" Condition="BasedOn"
Value="$ZCP270"/>
<QueryFilter Filter="Filter1" Condition="NamedLike" Va
lue="XYZ%"/>
<QueryFilter Filter="Filter1" Condition="!AssignedTo"
Value="Equip_Unit_001"/>
```

Notice the use of the keyword NamedLike in the Filter construction.

REPEAT

Direct Access can repeat a grouping of statements using the keywords:

- <PerformOperation> and </PerformOperation> to specify the boundary conditions of statements to be repeated.
- Repeat to specify the number of times the set of statements would be repeated.
- Start to specify the initial integer value when the repeat cycle begins. Note this value is incremented by 1 upon completing a cycle through the set of statements.
- Pattern to specify a character grouping that will be replaced by the value of the integer being incremented each pass through the set of statements.

Again, this is best understood by examining an example. Consider a group of repeated statements that perform the following operations:

- Create 100 controllers based on the FCP270 template
- Name the set of Controllers: A10000, A10100, A10200, …
- For each Controller, create 8 FBMs based on the FBM201 template
- Assign the currently created FBM to the currently created controller
- For Controller A10000, name that set of FBMs: A10001, A10002, …
- For Controller A10100, name that set of FBMs: A10101, A10102, …etc.
- For each Controller, create 8 FBMs based on the FBM204 template
- Assign the currently created FBM to the currently created controller

- For Controller A10000, name that set of FBMs: A10011, A10012, …
- For Controller A10100, name that set of FBMs: A10111, A10112, … etc.

The query statements to accomplish the above is constructed below.

```
<PerformOperation Repeat="100" Start="100" Pattern="ZZZ">
    <CreateController Template="$FCP270"
    Controller="AZZZ00"/>
    <PerformOperation Repeat="8" Start="1"
    RememberStart="Yes" Pattern="#">
    <CreateFBM Template="$FBM201" FBM="AZZZ0#"
    Controller="AZZZ00"/>
    </PerformOperation>
    <PerformOperation Repeat="8" Start="11"
    RememberStart="Yes" Pattern="??">
    <CreateFBM Template="$FBM204" FBM="AZZZ??"
    Controller="AZZZ00"/>
    </PerformOperation>
</PerformOperation>
```

Notice the pattern used for the creating the set of FBM270's is ZZZ and the initial value for ZZZ is 100; the pattern used for creating the set of FBM201's is ZZZ# and initial value of # is 1 while the pattern for creating the set of FBM204's is ZZZ?? and the initial value of ?? is 11. The names of both of the FBMs are created from the pattern governing the outer loop of the Controller which is ZZZ and the pattern governing the inner loop of their specific FBM which is # for FBM type 201 and ?? for FBM type 204.

MISCELLANEOUS COMMANDS

A variety of additional command utilities prove useful when managing large databases as indicated in the following examples.

- The SetVar command allows you to specify a variable and associated value to be used for string substitution in any command. You can use the variable in subsequent commands by bracketing its name with a set of percent signs (%). Whenever such a string is encountered on any command, the value of the variable will be used to replace the variable name, resulting in string substitution.
- The CommandFile command specifies a command file from which one or more additional commands are to be executed.
- The BackupGalaxy command allows you to back up the current Galaxy.
- The BulkCompile command allows you to bulk compile one or more control strategy templates or instances.
- The LogMessage command inserts an informational log message into the ArchestrA logger, viewable from the ArchestrA System Management Console (SMC).
- The Reset command allows you to reset, or change, the most recently referenced object. Many operations allow you to rely on the most recently referenced object, so that you do not need to re-enter it each time it's referenced. However, the Reset operation allows those most recently referenced objects to be changed, or nulled out.

HARDWARE AND SOFTWARE REQUIREMENTS

The Scripting with Direct Access is a component of Foxboro Evo Control Editors.

For detailed instructions on how to use Scripting with Direct Access, refer to the user document *Scripting with Direct Access User's Guide* (B0750BM).

For more information on the Control Editors, refer to the Product Specification Sheet *Control Editors* (PSS 31S-10EDITORS).

Foxboro®

by Schneider Electric

Invensys Systems, Inc
10900 Equity Drive
Houston, TX 77041
United States of America
http://www.invensys.com

Global Customer Support
Inside U.S.: 1-866-746-6477
Outside U.S.: 1-508-549-2424
Website: https://support.ips.invensys.com

Copyright 2014 Invensys Systems, Inc.
All rights reserved.
Invensys is now part of Schneider Electric.

Invensys, Foxboro, and Foxboro Evo are trademarks owned by Invensys Limited, its subsidiaries and affiliates. All other trademarks are the property of their respective owners.

MB 031 1214