Revision	Description
7/26/2010	Original

SQL-Hero Command Line Tool (SHCommand)

Introduction

There are inevitably cases where you would prefer to automate the process of deploying SQL change scripts to reconcile data and/or schema differences between databases, without requiring human intervention. One solution to this problem is to use the SHCommand.exe tool. We'll look at examples where this tool has been used to participate in a build process in a way that removes much effort that would have been required otherwise. Functionality has also been added to this tool to even invoke code generations that were previously only possible by building the .sqlheroproj project type under Visual Studio – meaning you can even further automate SQL-Hero processes.

The tool is command-line based and has the following usage:

$$\text{SHCommand config.SHCommand } [arg^1, arg^2, \ldots, arg^n]$$

A configuration file must *always* be used to provide details, such as the name of the SQL-Hero application server to use to access global connections by name, and at least one *task* to execute. The format of the config file is XML – we use parts of the XSD schema used to validate it below.

The optional arguments are used to replace substitution strings within the config file itself. These substitution strings look like those you find in the .NET String.Format() function: {0} for the first argument, {1} for the second argument, and so on.

Configuration File Format

The simplest possible configuration file would have this format:

```
<TASKS>
  <SERVER>servername</SERVER>
  <TASK action="actionname" level="detailtype"></TASK>
</TASKS>
```

We provide the SQL-Hero application server name (e.g. "localhost" if running all components locally). One or more tasks are listed in the order they should be executed. The "action" attribute can include the following:

Action	Description

Action	Description
BUILDFILTER	Builds a list of objects for use in subsequent tasks. It's possible to build the list by object type, objects changed in a certain timeframe, and/or by object name.
SCHEMACOMPARE	Compares schema objects between two databases.
SCHEMASCRIPT	Builds the change script that would be generated by running the prior SCHEMACOMPARE task. This does not apply it to the database, but allows you the opportunity to save it to a file.
SCHEMAROLLBACK	Builds the rollback script that would be generated by running the prior SCHEMACOMPARE task.
SCHEMAEXECUTE	Actually applies the change script that was built using the prior SCHEMASCRIPT.
UNITTEST	Not yet implemented – will eventually allow you to run SQL unit testing on demand
DATACOMPARE	Compares table data between two databases.
DATASCRIPT	Builds the change script that would be generated by running the prior DATACOMPARE task. This does not apply it to the database, but allows you the opportunity to save it to a file.
DATAEXECUTE	Actually applies the change script that was built using the prior DATASCRIPT.
SQLHEROGEN	Performs the code generation based on an input .sqlherogen file. This file type is associated with the SQL-Hero Build project type available under Visual Studio after SQL-Hero has been installed.
EMAIL	Sends e-mails to fixed distribution lists, supporting attachments.

The "level" attribute can be one of these values:

Level	Description
Summary	Presents only summary level information related to the task
SuccessPipeFormat	Depending on the task type, presents output in list format (e.g. list of object names) that can be fed into other tools
ErrorOnlyPipeFormat	Depending on the task type, presents output in list format for failure cases (e.g. list of object names that failed during the task) that can be fed into other tools
Detail	Presents detail level information related to the task
ErrorOnlyDetail	Presents detail level information related to the task, for failure cases only

Other optional TASKS level elements include:

```
<xs:element name="PREVIEW" type="xs:boolean" minOccurs="0" />
<xs:element name="SILENT" type="xs:boolean" minOccurs="0" />
<xs:element name="FULL_QUALIFIED_NAMES" type="xs:boolean" minOccurs="0" />
```

PREVIEW can be "true" or "false" and if true, scripts are not actually executed in target databases, but all other aspects "work" so you could for example get a preview of the script file that *would* be applied,

before actually applying it. SILENT can be "true" or "false" as well and when true, suppresses *all* output from the tool. FULL_QUALIFIED_NAMES is also a Boolean (default is "false"), and when "true", always deals in two-part names (i.e. includes the schema).

Task Types

This section details each task type, including the child elements that are available for use. The format we've used is the XSD which is used to validate the .SHCommand XML document – this tells you the data type expected, plus whether the element is required or not (minOccurs="0" implies it's optional).

BUILDFILTER

The BUILDFILTER task constructs a list of objects that are stored internally for later use in subsequent tasks.

```
<xs:element name="IN" type="xs:string" minOccurs="0" maxOccurs="1" />
```

"IN" names the database connection that would be used if using the CHANGED_SINCE option, or other options that rely on knowing about the contents of a specific database in order to build the object list.

```
<xs:element name="OBJTYPE" type="xs:string" minOccurs="0" maxOccurs="1" />
```

"OBJTYPE" is an optional comma separated list of object type codes. These codes correspond to standard SQL Server object types (e.g. "U" for table, "P" for stored procedure, etc.).

```
<xs:element name="CHANGED_SINCE" type="xs:string" minOccurs="0" maxOccurs="1" />
```

"CHANGED_SINCE" allows you to filter only objects changed since a certain date/time, determined by a time span specified in this element. The time span is subtracted from the current date/time. The time span should be a number followed by one of these letters: "h" for hours, "d" for days, "w" for weeks.

```
<xs:element name="OBJECTS" type="xs:string" minOccurs="0" maxOccurs="1" />
```

"OBJECTS" can be used to explicitly list the objects to participate in the filter. The list can be carriage return or comma delimited.

```
<xs:element name="COMPARE_FOR_DELETE" type="xs:string" minOccurs="0" maxOccurs="1" />
```

When *not* specified, deletions of objects in the target database will not be possible. This is because as you build a list against a source database, if objects are missing from the source, they'll be missing from the constructed object list filter and so are in effect, invisible for the purposes of comparison. Specifying a database alias in this element enables deletions since the object name list is unioned with objects in both the source and target databases.

```
<xs:element name="ListManager" minOccurs="0" maxOccurs="1">
	<xs:element name="moModAfter" type="xs:string" minOccurs="0" maxOccurs="1"/>
	<xs:element name="moModBefore" type="xs:string" minOccurs="0" maxOccurs="1"/>
	<xs:element name="mblnNoHideIfParentVisible" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
	<xs:element name="mblnNoHideIfChildVisible" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
	<xs:element name="mblnExactMatching" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
	<xs:element name="msModBy" type="xs:string" minOccurs="0" maxOccurs="1"/>
	<xs:element name="mbNotModBy" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
```

Use of the ListManager element and its child elements gives a finer level of control over the filter, but specific object names cannot be used here.

Example:

```
  <TASK action="BuildFilter" level="Summary">
    <IN>RD-Development</IN>
    <CHANGED_SINCE>3d</CHANGED_SINCE>
    <COMPARE_FOR_DELETE>RD-QA</COMPARE_FOR_DELETE>
  </TASK>
```

SCHEMACOMPARE

The SCHEMACOMPARE task builds the list of differences in schema objects between two databases that can be used in subsequent tasks.

```
<xs:element name="FROM" type="xs:string"/>
```

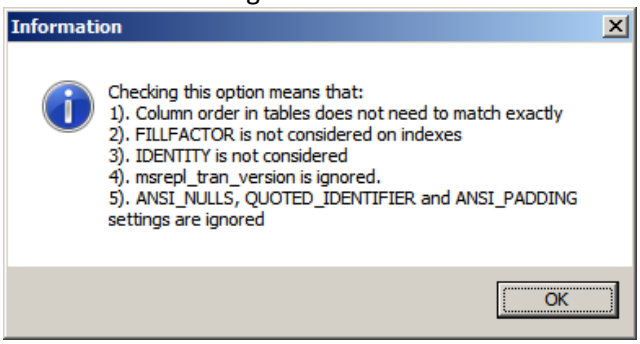
The “FROM” element names the global connection alias of the database that objects are being moved *from*.

```
<xs:element name="TO" type="xs:string"/>
```

The “TO” element names the global connection alias of the database that objects are being moved *to*.

```
<xs:element name="RELAXED_MATCHING" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

The relaxed matching semantics that are listed here are used (default is “true”):




```
<xs:element name="INCLUDE_TESTS" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When “true”, unit tests for SQL objects are included in any future scripting task. Default is “false”.

```
<xs:element name="IGNORE_CASE" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When “true”, comparisons are case insensitive. Default is “true”.

```
<xs:element name="MATCH_TYPE" type="xs:string" minOccurs="0" maxOccurs="1" />
```

Allows one to restrict the comparison to one of: “`InSourceNotTarget`”, “`InTargetNotSource`”, “`Different`”. If the MATCH_TYPE element is missing, any difference is included.

Example:

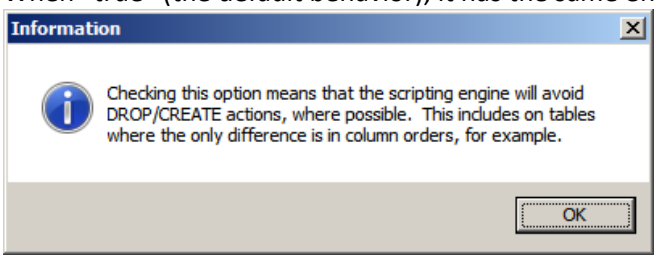
```
<TASK action="SchemaCompare" level="Detail">
  <FROM>RD-Development</FROM>
  <TO>RD-QA</TO>
</TASK>
```

SCHEMASCRIPT

This task requires that a SCHEMACOMPARE task has run previously. This does not actually apply the schema change script, simply builds it and allows it to be written to a file target.

```
<xs:element name="RESTRICTIVE_SCRIPTING" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When “true” (the default behavior), it has the same effect as described here, within the application:




```
<xs:element name="TARGET" nillable="true" minOccurs="0" maxOccurs="1">
      <xs:extension base="xs:string">
            <xs:attribute name="type" type="xs:string" />
```

The TARGET optional element allows you to name a target for the output of the script. Certain substitution tags are available, as well. These include [date:] and [username], as illustrated in the example below. Target types include: “File” or “Console”. You can only supply one TARGET element for a given task.

Example:

```
<TASK action="SchemaScript" level="Detail">
  <TARGET type="File">ScriptDevToQA[date:yyyyMMddHHmm][username].sql</TARGET>
</TASK>
```

SCHEMAROLLBACK

Similar to the SCHEMASCRIPT task, this causes output to a target – the output in this case is the rollback script necessary to undo the effect of applying the generated change script from SCHEMASCRIPT. The same options exist as for SCHEMASCRIPT.

SCHEMAEXECUTE

This task actually executes the SQL from a script previously generated by a SCHEMASCRIPT task. Note that when the PREVIEW mode is set to "true" for the session, this task will still attempt to apply schema changes, but it rolls these changes back, meaning the output of this task can still have meaning even in preview mode.

```
<xs:element name="IN" type="xs:string"/>
```

The "IN" element names the global database alias which the generated script will be executed in.

```
<xs:element name="STOP" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When "true", the STOP element causes script execution to stop immediately when an error is encountered. The alternative is that the script continues to execute, potentially applying changes for other objects. Default is "false".

```
<xs:element name="OUTPUT" type="xs:string" minOccurs="0" maxOccurs="1" />
```

As opposed to the TARGET output reflecting what is actually being executed by this task, the OUTPUT element captures error information which is generated from SQL Server as the schema change script is executed. This therefore includes error messages which may result. The element is a filename, and can include the substitution strings described for SCHEMASCRIPT.

```
<xs:element name="TIMEOUT" type="xs:integer" minOccurs="0" maxOccurs="1" />
```

The "TIMEOUT" element provides a timeout (in milliseconds) that is applied to each command as they execute in the generated change script. The default is 60 seconds.

```
<xs:element name="FIXME_HANDLING" type="xs:string" minOccurs="0" maxOccurs="1" />
```

In some cases, SQL-Hero cannot effectively produce a change script without requiring developer intervention. Normally when using the GUI, the generated script includes the text "FIXME!" which lets the developer quickly see where they need to typically provide default values. Since the command line tool is intended to be completely automated, this option lets you define how cases where "FIXME!" would appear are dealt with. These options exist:

Option	Description
`Continue`	Still execute the generated statement. It will inevitably fail, but this will be logged. This is the default.
`ErrorFullTask`	The entire task is failed – no statements will be run if a FIXME! is detected anywhere in the generated script.

Option	Description
SkipStatement	Statements with FIXME! are simply skipped – a warning is provided.
TryUseZero	FIXME! text will be replaced with "0" which will work in many cases – this implies that new required columns would receive a value of "0" regardless of what the real values should be.

Example:

```
<TASK action="SchemaExecute" level="Summary">
  <IN>RD-QA</IN>
  <OUTPUT>ScriptDevToQA[date:yyyyMMddHHmm][username].log</OUTPUT>
  <FIXME_HANDLING>SkipStatement</FIXME_HANDLING>
</TASK>
```

DATACOMPARE

The DATACOMPARE task builds the list of differences in data between two databases that can be used in subsequent tasks.

```
<xs:element name="FROM" type="xs:string"/>
```

The "FROM" element names the global connection alias of the database that objects are being moved *from*.

```
<xs:element name="TO" type="xs:string"/>
```

The "TO" element names the global connection alias of the database that objects are being moved *to*.

Example:

```
<TASK action="DataCompare" level="Detail">
  <FROM>RD-Development</FROM>
  <TO>RD-QA</TO>
</TASK>
```

DATASCRIPT

This task requires that a DATACOMPARE task has run previously. This does not actually apply the data change script, simply builds it and allows it to be written to a file target.

```
<xs:element name="SUPPRESS_TRIGGERS" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

The SUPPRESS_TRIGGERS element, when "true" causes triggers to be disabled on target tables which can be useful when dealing with certain auditing scenarios. Default is false.

```
<xs:element name="DELETE_CHILDREN" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When “true”, DELETE_CHILDREN will remove child records when a delete is scripted for a parent record. When “false”, the delete of the parent will fail. Default is true.

```
<xs:element name="SET_NULL" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When “true”, the preference will be to null out a child reference when a parent would be deleted. This is only possible for cases where the parent key is nullable on the child table. Default is true.

```
<xs:element name="TARGET" nillable="true" minOccurs="0" maxOccurs="1">
       <xs:extension base="xs:string">
              <xs:attribute name="type" type="xs:string" />
```

The TARGET optional element allows you to name a target for the output of the script. Certain substitution tags are available, as well. These include [date:] and [username], as illustrated in the example below. Target types include: “File” or “Console”. You can only supply one TARGET element for a given task.

Example:

```
<TASK action="DataScript" level="Detail">
  <TARGET type="File">DataDevToQA[date:yyyyMMddHHmm][username].sql</TARGET>
</TASK>
```

DATAEXECUTE

This task actually executes the SQL from a script previously generated by a DATASCRIPT task. Note that when the PREVIEW mode is set to “true” for the session, this task will still attempt to apply data changes, but it rolls these changes back, meaning the output of this task can still have meaning even in preview mode.

```
<xs:element name="IN" type="xs:string"/>
```

The “IN” element names the global database alias which the generated script will be executed in.

```
<xs:element name="STOP" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

When “true”, the STOP element causes script execution to stop immediately when an error is encountered. The alternative is that the script continues to execute, potentially applying changes for other records. Default is “false”.

```
<xs:element name="OUTPUT" type="xs:string" minOccurs="0" maxOccurs="1" />
```

As opposed to the TARGET output reflecting what is actually being executed by this task, the OUTPUT element captures error information which is generated from SQL Server as the data change script is executed. This therefore includes error messages which may result. The element is a filename, and can include the substitution strings described for SCHEMASCRIPT.

```
<xs:element name="TIMEOUT" type="xs:integer" minOccurs="0" maxOccurs="1" />
```

The "TIMEOUT" element provides a timeout (in milliseconds) that is applied to each command as they execute in the generated change script. The default is 60 seconds.

Example:

```
<TASK action="DataExecute" level="Detail">
  <IN>RD-QA</IN>
  <OUTPUT>DataDevToQA[date:yyyyMMddHHmm][username].log</OUTPUT>
</TASK>
```

SQLHEROGEN

This task type essentially runs a named .sqlherogen file as if it were being built under Visual Studio as part of a .sqlheroproj project. This is described in some detail here.

```
<xs:element name="FILE" type="xs:string"/>
```

The FILE element is used to name the .sqlherogen file that contains the list of templates which will be executed. The easiest way to build a .sqlherogen file is to use Visual Studio's designer tool associated with that file type.

Example:

```
<TASK action="SQLHeroGen" level="Detail">
  <FILE>c:\projects\rddev\builditem.sqlherogen</FILE>
</TASK>
```

EMAIL

The email task uses a connection to a named SMTP server to send email that can include file attachments that were generated as part of the scripting process.

```
<xs:element name="MAILSERVER" type="xs:string"/>
```

The SMTP mail server name is required.

```
<xs:element name="FROM" type="xs:string" minOccurs="0" maxOccurs="1" />
```

The FROM element provides that name that shows up as the sender of the email. The default is "Automation@SQLHero.null".

```
<xs:element name="TO" minOccurs="1" maxOccurs="unbounded">
```

One or more TO elements list the email address of recipients of the message.

```
<xs:element name="SUBJECT" type="xs:string" minOccurs="0" maxOccurs="1" />
```

The SUBJECT element inner text becomes the subject for the outgoing message. The default subject is "SQL-Hero command line tool email".

```
<xs:element name="BODY" type="xs:string" minOccurs="0" maxOccurs="1" />
```

The BODY element inner text becomes the body for the outgoing message. The default body is empty.

```
<xs:element name="ATTACHMENT" minOccurs="0" maxOccurs="unbounded">
```

The ATTACHMENT element inner text can name a file that should be attached to the outgoing message. The filename can include substitution strings that are described under SCHEMASCRIPT.

Example:

```
  <TASK action="EMail" level="Detail">
    <MAILSERVER>smptserver</MAILSERVER>
    <TO>fred@codexframework.com</TO>
    <FROM>SHSyncTool@SH.Null</FROM>
    <SUBJECT>Command line tool completed</SUBJECT>
    <BODY>The command line tool completed.  Attached are the generated scripts which were
run.</BODY>
    <ATTACHMENT>ScriptDevToQA[date:yyyyMMddHHmm][username].sql</ATTACHMENT>
    <ATTACHMENT>ScriptDevToQA[date:yyyyMMddHHmm][username].log</ATTACHMENT>
    <ATTACHMENT>DataDevToQA[date:yyyyMMddHHmm][username].sql</ATTACHMENT>
    <ATTACHMENT>DataDevToQA[date:yyyyMMddHHmm][username].log</ATTACHMENT>
  </TASK>
```