

timeXtender®



What's New in timeXtender 4.5

What's New in timeXtender 4.5

- New Features and Important Notes

by Thomas Christiansen, CTO timeXtender

Contents

What's New in timeXtender 4.5	1
Important Upgrade Notes.	1
History Enabled Projects (and Tables)	1
Project Type "Versioning"	1
Storage Optimized (and Simplified) Table Structure	1
Execution Packages.	1
Lookup Fields.	2
User Interface Changes	2
Show Advanced Features.	2
Hashed Keys	2
Input Limit on Hash Fields	2
Storage and Deterministic Requirements	3
Hash Value for Primary Key (Natural Key)	3
User-Defined Functions	4
Performance Gains	4
Summary.	4
Partitioning - Tables	4
Partitioning - Incremental Update of Cubes.	4
Compression	5
Lookup Fields - Grouping.	5
Valid Table Instance - Physical Table or View	5
Add Related Records	5
Slowly Changing Dimensions Support – Optimized	6
Target Schema	7
Configuration of Table Schemas	7
Deployment of Schemas	7
Table Feature Matrix	8
Combination of Source and Target-Based Incremental	8
Default Settings for New Objects	9
Tables	9
Data Quality	10
Primary Key Behavior	10
Integration	11
Executing External SSIS Packages.	11
Command Line Execution	11

What's New in timeXtender 4.5

- Important Upgrade Notes

The following changes affect upgraded projects.

■ History Enabled Projects (and Tables)

To deploy history enabled projects in version 4.5, you need to manually convert all tables into the new implementation. This means that while you will be able to execute these projects as normal after upgrade, you will not be able to (re)deploy any object until the manual conversion has been completed.

The support for slowly changing dimensions (SCD) has been improved significantly and comes with fewer limitations and a much clearer user interface. We therefore strongly recommend reading the section on this topic prior to conversion.

■ Project Type "Versioning"

This project type is no longer supported since it is not in line with the industry best practices for performance optimized solutions. You will be able to open projects and convert them into standard projects. The alternative to versioning is to implement slowly changing dimensions (SCD) as a way of storing multiple instances of data.

■ Storage Optimized (and Simplified) Table Structure

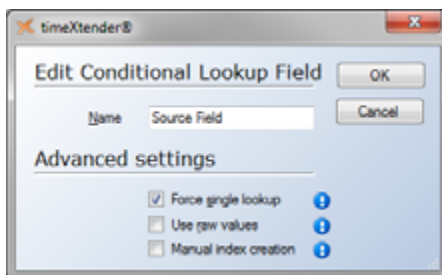
To minimize storage requirements, we have restructured the handling errors and warnings generated by the data quality process. As a result the error post fixed with "_E" for errors and "_W" for warnings will no longer be deployed. For upgraded projects this means that the tables will no longer be implemented on SQL Server. Existing tables will have to be deleted manually to ensure that no custom code is broken by automatic deletion.

■ Execution Packages

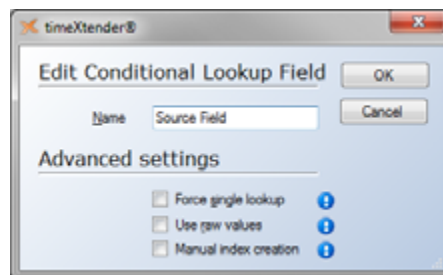
It is no longer possible to create new execution packages in the legacy format that was deprecated in timeXtender version 4.2. It will still be possible to edit and execute execution packages created in earlier versions of timeXtender, but we encourage you to restructure your packages into the current package format.

■ Lookup Fields

Performance on lookup fields has been greatly improved in this version including among other things grouping of fields. To avoid unexpected results on upgraded projects, all lookup fields in existing projects will act as in the previous version, and will therefore not automatically take advantage of the new grouping behavior (please see elsewhere in this document for further details). The existing lookup fields have to be manually configured like illustrated in the following to enable automatic grouping.



Settings on upgraded lookup field



Settings on grouping enabled lookup field

User Interface Changes

■ Show Advanced Features

On the Tools menu, Window and Menu settings allow you to switch display of advanced features on and off. It is therefore possible to hide some of the more complex features for business users with less technical expertise, who are unlikely to use the features. This setting can be configured on a per user basis.

Hashed Keys

One of the major changes in this version is the introduction of single-field binary keys that greatly improve performance in join and comparison operations. The fields are populated as computed columns, using the `HASHBYTES` function with the `SHA1` algorithm, which is considered safe in terms of key collision.

■ Input Limit on Hash Fields

The input for the hash function will be converted to a single string value with a maximum length of 8000 characters. If the actual length of the concatenated string value exceeds the limitation, it is truncated at the maximum length.

There is no warning or error produced by neither timeXtender nor SQL Server in such case since it is considered an unlikely scenario not worth the performance costs. Any such truncation means that if there are multiple records in a table with the same value on the first 8000 characters the hashed value will not be unique.

■ Storage and Deterministic Requirements

To be able to persist the hashed values for best performance, some transformations using custom SQL may not be possible or have special requirements. Please refer to SQL Server Books Online for “deterministic and nondeterministic functions” for the full details. The functions known at the time of writing this document are **CAST**, **CONVERT**, **CHECKSUM**, **ISDATE** and **RAND**.

Any date time based field used as the input to a system generated hash value in timeXtender will be converted using the fixed style 101 to ensure the convert function is deterministic. This includes the hash values generated for primary keys, natural keys, type I and II dimension keys.

■ Hash Value for Primary Key (Natural Key)

All fields used as primary key will be hashed as a single value in the **BK Hash Key** field. This single binary field will be used for all joins that involve the primary key. If no primary key has been defined on a table, the surrogate key field **DW_Id** will be used as hash value for consistency.

In the following example, the primary key fields HotelID and RoomNo are concatenated and hashed into the system field **BK Hash Key**, persisted as a computed column on the table.

HotelID	RoomNo	DW_Id	DW_Batch	DW_SourceCode	DW_TimeStamp	BK Hash Key
5001	101	1	3	happy	2010-09-27	0x8F807193FFFC70948A60A305F9606D87849145C
5001	102	2	3	happy	2010-09-27	0xB2170387DF0786466C95E24373E30EE2C0C882F
5001	103	3	3	happy	2010-09-27	0xC4271AE771D54E2970881A426427F33DF1588F08
5001	104 - Blue Sky	4	3	happy	2010-09-27	0x17517784936D812478722A19654075AD7AC703F
5001	105	5	3	happy	2010-09-27	0xB99A571D890C7342895885F74743485426101
5001	106	6	3	happy	2010-09-27	0x7824683C058D252453B4041A82594451006216
5001	107	7	3	happy	2010-09-27	0xE07CE78B9F14D5349531144FC5FFD4A779F2D5F5
5001	108	8	3	happy	2010-09-27	0x2C06E256D5CF8437F406DA8F73E32F10201A359A
5001	109	9	3	happy	2010-09-27	0xB3E19AAFAE238E4CAB86AC9C0C8FAC7FE50E5
5001	110	10	3	happy	2010-09-27	0x5A52526EEDF570430751A63185335A793C8D726
5001	111	11	3	happy	2010-09-27	0x64CA98E34947C954C3ADD0E7A19CA33D8CC0611
5001	112	12	3	happy	2010-09-27	0xF4A63F8FE8677837E73D4119E162E78878AD42

User-Defined Functions

User-defined functions will by default use the `WITH SCHEMABINDING` option to improve performance. In summary, the `SCHEMABINDING` option should be specified for T-SQL UDFs to ensure that plans using these UDFs do not cause unnecessary spools.

Making the UDF schema-bound, forces SQL Engine to analyze the body of the UDF and properly set derived properties. You will avoid unnecessary spooling and may see a very significant performance gain.

The schema binding option is specified in the table settings dialog for a table.

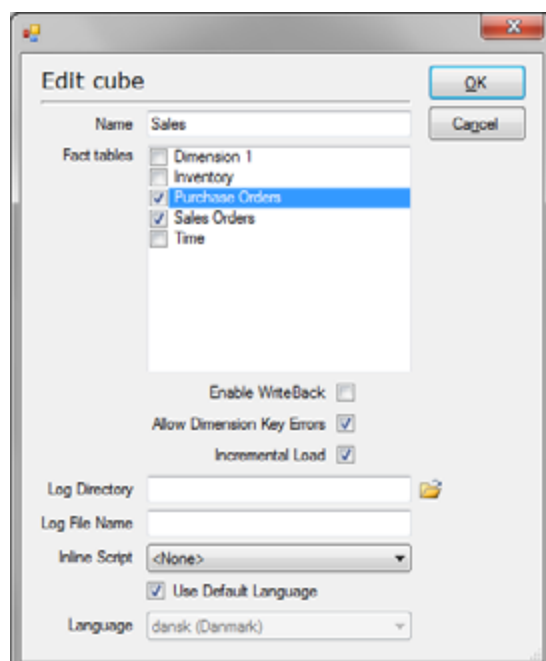
Performance Gains

■ Summary

Errors and warnings are no longer handled in separate tables. This results in both storage saving and in removing the two processing steps from the overall process. The impact of this is depends on the number of errors and warnings of an execution. The performance gain of these changes will increase with the number of errors and warnings, and have little effect on the overall performance in situations with few errors and warnings.

■ Partitioning - Tables

When deploying to SQL Server 2008+ Enterprise Edition, you now have the ability to use table partitioning on stage- and data warehouse databases. This is implemented as partition templates that can be used across multiple tables. Partitioning is often done on a date based value such as a year-month combination, but can also be done on numeric or text based values.



■ Partitioning - Incremental Update of Cubes

When a partitioned table on the data warehouse is used as a fact table on a cube, fact table partitions are automatically created and aligned with the partitions on the table. This will increase cube processing performance in both full and incremental scenarios, especially on machines with multiple processors/cores.

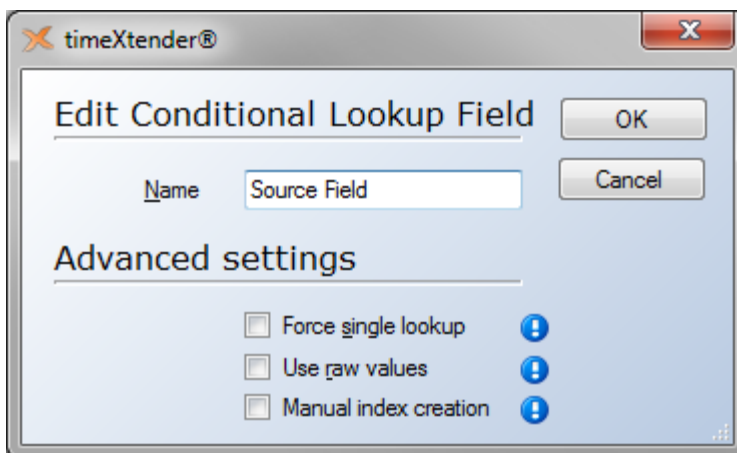
On the cube definition, a new parameter allows the enabling of automatic incremental update. When this setting is enabled, timeXtender will examine when the last timestamp data in a table partition has been inserted or updated and compare this to the last processing timestamp of cube partitions and process partitions accordingly. Assuming that data is being loaded incrementally into the data warehouse and partitioned on a suitable field, this can increase cube processing by as much as 80-90%.

■ Compression

The new table classification parameter on table settings controls if and how a table uses data compression. Any *large classification will implement compression when available on the target environment (depending on license, SQL Server edition and SQL Server version). Tables will be compressed according to the following table.

Destination	Compression Type	Storage Saving	Performance Gain
Staging	Row	40-60%	0-20%
Data Warehouse	Page	50-90%	30-60%

■ Lookup Fields - Grouping



Automatic creation of non-clustered index for fields will be created by the table performing the lookup. For example, if a lookup is done on table Sales Order Details to fetch a value on table Sales Order Header, the data cleansing process for Sales Order Details will drop- and create indexes on the Sales Order Header table as needed.

■ Valid Table Instance - Physical Table or View

As a new way of optimizing both storage and process time, it is now possible to choose to have the valid table instance implemented as a view, whereas previous versions would always implement it as a physical table. This option can be set on a per table basis using the Advanced Settings dialog. In some scenarios it will not be possible to use this option, as a physical implementation of the valid table instance is required. This includes history enabled- and target-based incremental tables.

■ Add Related Records

Since version 2005, SQL Server has supported the ability to convert missing values into unknown members on dimensions. Although this is a very powerful feature to ensure consistency in data, it comes with a price of slow performance if there is a high volume of missing values that needs conversions. One solution to this is to create derived dimension records in the relational data model in the staging- and data warehouse databases. This allows, for example, the creation of dimension keys from a fact table that does not exist in a related dimension table. In addition to improving the overall load performance, it also enables valuable reporting capabilities to the relational data.

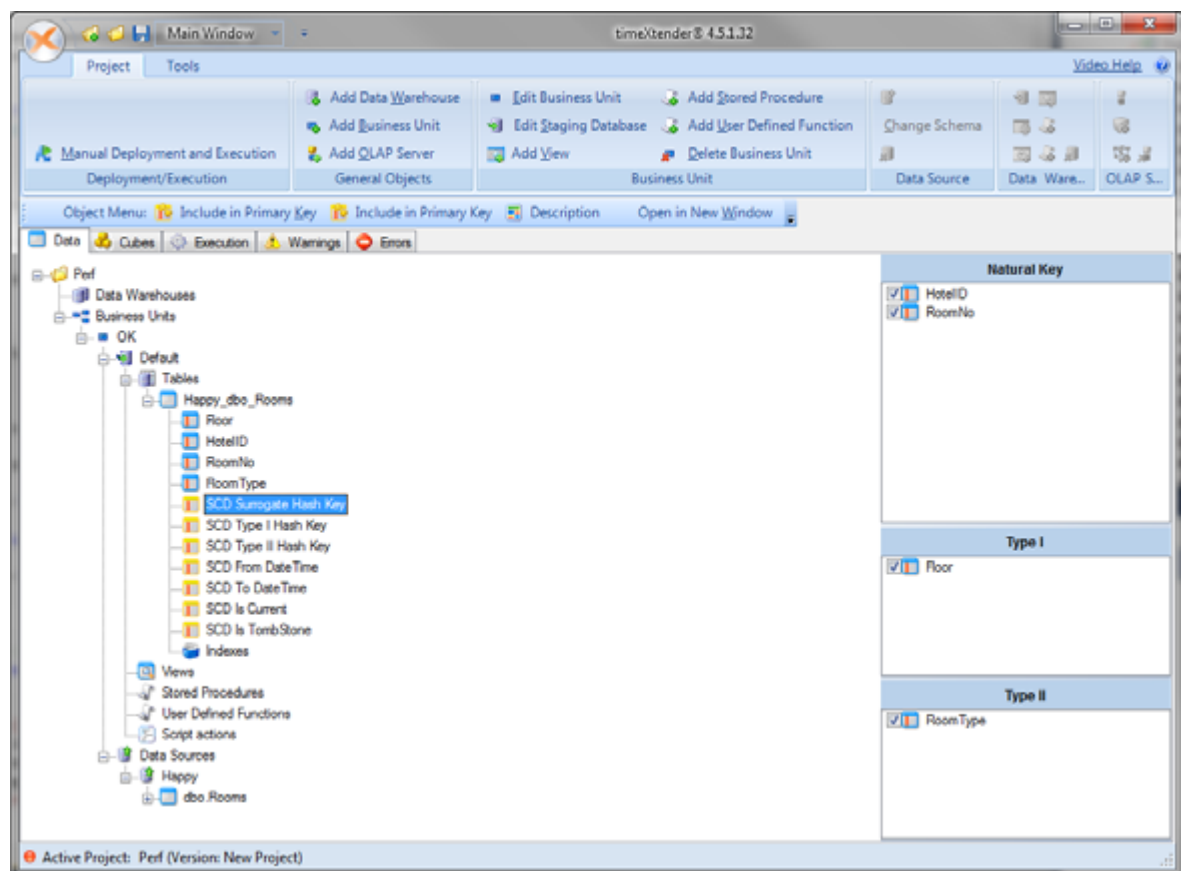
Slowly Changing Dimensions Support - Optimized

The support for especially type 2 dimension history has been refactored for simplified configuration, more flexibility, and better performance. Every field in a table with history enabled, needs to be configured as either part of the natural key (business key), a type 1 or type 2 attribute.

Natural key: Used to identify the logical instance of an object. This will often be equal to the primary key in the source system.

Type 1 attribute: Identifies values that are not significant for tracking history, thus any changes to these values will be updated on all historical records with no specific tracking of when the change occurred. Changes in type 1 attributes will not create new instances of records.

Type 2 attribute: Used to identify values that are tracked for historical changes. Any changes in these attributes will be compared to the current (latest) instance identified by the natural key. If any of the type 2 attributes have changed, a new instance will be created. The from and to date/time values on the records will identify when any record was active in terms of time.



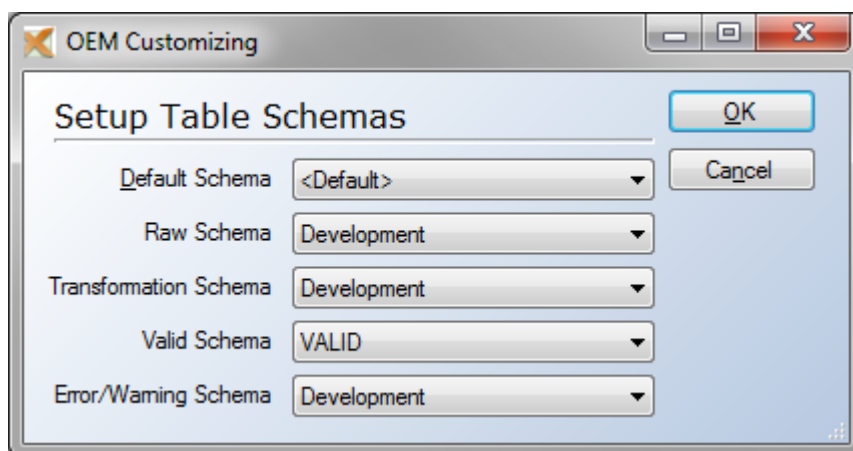
Target Schema

Support for schemas has been extended to also include staging and data warehouse databases. Schemas are defined per database and one schema can be marked as the database default schema.

Schemas can be used to manage access to data on the relational database, but this will have to be maintained in SQL Server native tools. To prevent unintended overwriting of schemas, a deployment guard can be added for protection.

■ Configuration of Table Schemas

Instance	Objects Affected
Default	Used for all instances that have no schema specified. Used for user-defined views
Raw	The raw table + incremental _I/_Incr + "ado.net views"
Transformation	_T view + associated UDF + SP + Custom Data + mapping
Valid	Valid table + write back
Error/Warning	Tables _L + _M



■ Deployment of Schemas

When deploying objects on SQL Server, the above schemas will be used according to this fallback strategy:

1. Use specific schema defined on table instance.
2. If not specified, use default schema on table.
3. If default schema on table is not specified, use default schema for database.
4. If no default schema exists on database, use application default schema.
5. Application default schema = dbo, owned by dbo (fixed value).

Schemas are only be physically deployed onto SQL Server if they do not exist. This is handled in the SQL statement with the following syntax.

Table Feature Matrix

The default load strategy for any table in timeXtender is called full load, which means that all records are extracted, transformed and loaded from the source into its destination. This default data flow executes in the following main steps

1. Delete all records from the raw table.
2. Insert all data from the source into the raw table.
3. Perform data cleansing on all records in the raw table.
4. Perform data quality checks.
5. Delete all records from the valid table.
6. Copy all valid records into the valid table.

This flow can be adjusted towards specific needs like performance optimization or tracking dimensional changes according to the following matrix of allowed combinations.

	Static Selection Rules	Source Based Incremental	Target Based Incremental	History I + U	History I + U + D	Truncate Raw	Truncate Valid
Static Selection Rules		X	X	X	X	X	X
Source Based Incremental	X		X	X	-	X	-
Target Based Incremental	X	X		-	-	X	-
History I + U	X	X	-		X	X	-
History I + U + D	X	-	-	X		X	-
Truncate Raw	X	X	X	X	X		X
Truncate Valid	X	X	-	-	-	X	

■ Combination of Source and Target-Based Incremental

If both source and target based incremental loading are added to a table, the source-based will determine how data is being extracted from the source, while the target-based will determine how data is inserted, updated or deleted in the destination table (the Valid table instance).

When delete events are enabled in this combination, it effectively means that the tables will only contain the records from the last execution. Although this is unlikely to be the typical usage, this could be useful for the staging database in some scenarios.

Default Settings for New Objects

■ Tables

The following describes the default parameter when a new table is created. Please be aware that the settings marked with * can be overwritten by a default setting on the project.

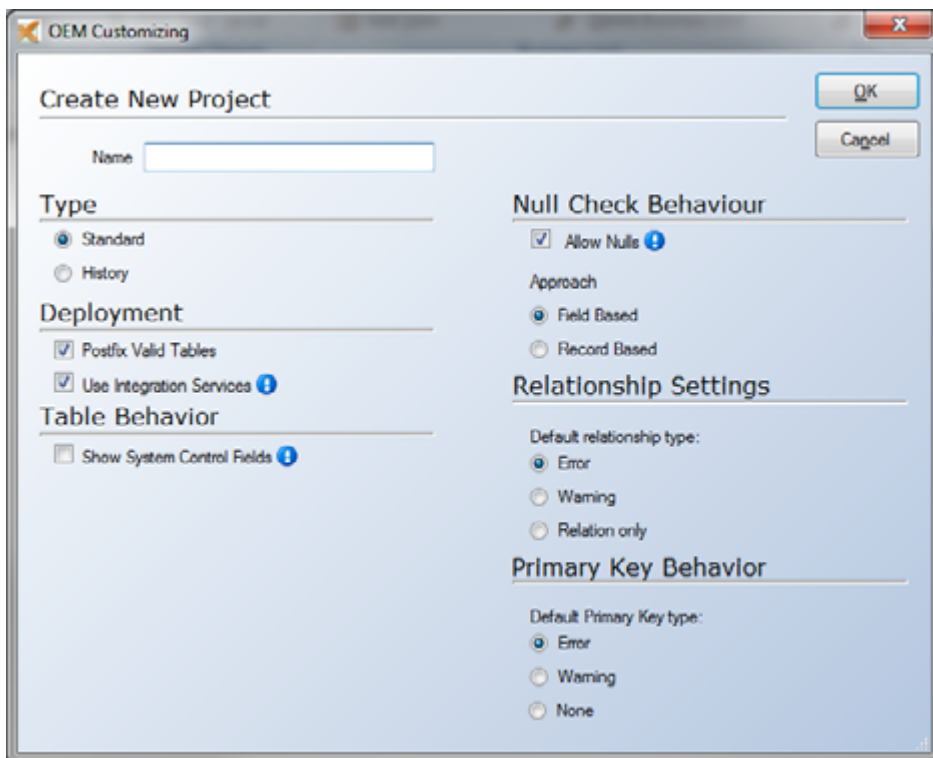
Setting	Staging	Data Warehouse	Consolidation	Time
Table classification	Dimension	Dimension	Dimension	Dimension
Primary key behavior *	None	None	None	None
Partitioning	None	None	None	None
Show system fields *	No	No	No	No
Incremental Loading				
Source based	No	No	N/A	N/A
Target based	No	No	N/A	N/A
History Tracking				
Enabled	No	No	No	N/A
Type I - Update all	No	No	No	N/A
Handle deletes	No	No	No	N/A
Truncation				
Raw before transfer	Yes	Yes	Yes	Yes
Valid before transfer	Yes	Yes	Yes	Yes
Raw after data cleansing	No	No	No	No
Performance				
Physical valid table	Yes	Yes	Yes	Yes
Schema bind functions	Yes	Yes	Yes	Yes

Data Quality

■ Primary Key Behavior

In previous versions of timeXtender, primary keys were always enforcing the primary key. As a result, records violating the primary key uniqueness constraint were set aside in the errors table. A new option now makes it possible to consider uniqueness violations as warnings or ignore them completely. This can be defined as a default behavior at project level and optionally overridden on a per table level.

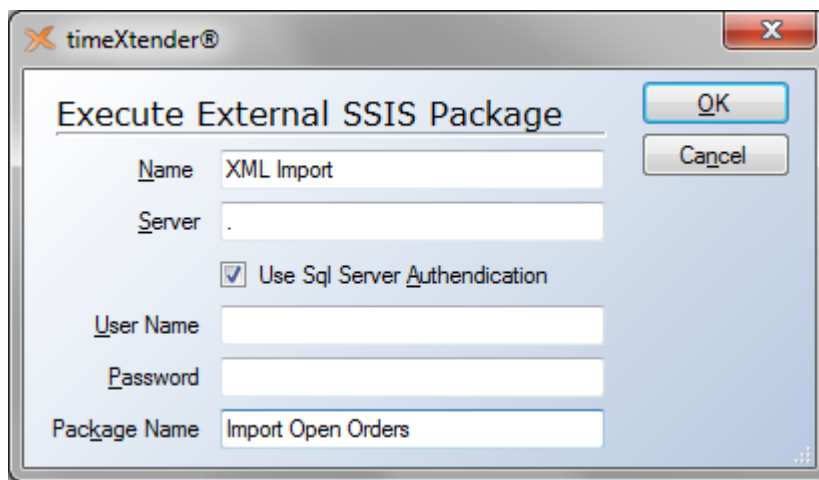
New- and upgraded projects will default to a behavior of Error as shown in the following.



Integration

■ Executing External SSIS Packages

A new feature makes it possible to execute SSIS packages outside timeXtender. Packages stored in the SSIS repository are supported, and there is currently no support for passing parameters to the packages. The external packages can be executed as an integrated part of a normal execution package.



■ Command Line Execution

If you execute timeXtender from a command-line or using a third party scheduler, the following return values can be used to determine if an execution has completed with success or in error. The syntax for executing timeXtender from a command line call is

```
timeXtender.exe "project name" "execution package name"
```

Return values are

- 1 Execution failed; please refer either to the trace log file or Windows event log for details
- 0 Project or execution package was not found
- > 0 A positive integer value indicates a successful execution, referring to the batch ID that was assigned. Please refer to table ExecutionLogs for details about execution times.

Disclaimer

This document is provided for informational purposes only, and is subject to change without notice. The information contained in this document represents the current view of timeXtender on the issues discussed as of the date of publication. This document should not be interpreted to be a commitment on the part of timeXtender, and timeXtender cannot guarantee the accuracy of any information presented after the date of publication. This document is for informational purposes only. timeXtender makes no warranties, express, implied, or statutory, with respect to the information in this document. Complying with all applicable copyright laws is the responsibility of the user. No part of this document may be reproduced or transmitted in any form without written permission from timeXtender.

Copyright

timeXtender® is a registered trademark of timeXtender Holding. Microsoft®, Microsoft® Dynamics®, and SQL Server® are either registered trademarks or trademarks of Microsoft Corporation, or Microsoft Business Solutions ApS in the United States and/or other countries. All other product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

© 2011 timeXtender as. All rights reserved.