[MS-DTS]:
Data Transformation Services Package File Format Specification

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
07/07/2011	0.1	New	Released new document.
11/03/2011	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/19/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the Data Transformation Services (DTS) package file format, which is the file format that is used to write and read from a DTS **package** file. The DTS package file format is used to persist DTS packages in the Microsoft Windows® file system.

Sections 1.7 and 2 of this specification are normative and contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

globally unique identifier (GUID)
little-endian
salt
Universal Naming Convention (UNC)

The following terms are specific to this document:

compound file: A file that is created as defined in [MS-CFB] and that is capable of storing data that is structured as **storage** and **streams**.

DTS constraint: A means of controlling the workflow within the DTS **package**. For more information, see [MSDN-DTSPW].

DTS task: A discrete functionality that is used by the DTS **package** file format. For more information, see [MSDN-DTSTAS].

DTS transformation: One or more operations to apply to data that is in transit from one location to another. For more information, see [MSDN-DTSTRANS].

package: A collection of connections, tasks, transformations, and **DTS constraints** that perform a particular operation or set of operations, as described in [MSDN-DTSBAS]. There can be multiple **versions** of one collection within a package.

root storage object: The top-level **storage** object in a **compound file**, as defined in [MS-CFB].

storage: A storage object, as defined in [MS-CFB].

stream: A stream object, as defined in [MS-CFB].

substorage: A **storage** object in a **compound file** that is contained within another storage object or the **root storage object**.

version: A saved collection within a **package**. Each time a new collection is saved into a package, that new collection becomes the default version that is used by the package.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-CFB] Microsoft Corporation, "Compound File Binary File Format".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol Specification".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

1.2.2 Informative References

[MSKB229884] Microsoft Corporation, "How To Use OLE DB DBTYPE_VARNUMERIC", http://support.microsoft.com/kb/229884

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MSDN-BCPU] Microsoft Corporation, "bcp Utility", http://msdn.microsoft.com/en-us/library/ms162802(SQL.105).aspx

[MSDN-COLID] Microsoft Corporation, "Column IDs", http://msdn.microsoft.com/en-us/library/ms709735%28v=VS.85%29.aspx

[MSDN-DTS] Microsoft Corporation, "Data Transformation Services (DTS)", http://msdn.microsoft.com/en-us/library/cc707786.aspx

[MSDN-DTSBAS] Microsoft Corporation, "DTS Basics", http://msdn.microsoft.com/en-us/library/aa933485%28SQL.80%29.aspx

[MSDN-DTSPW] Microsoft Corporation, "DTS Package Workflow", http://msdn.microsoft.com/en-us/library/aa933535(v=SQL.80).aspx

[MSDN-DTSS2008R2] Microsoft Corporation, "Support for SQL Server 2000 DTS in SQL Server 2008 R2", http://msdn.microsoft.com/en-us/library/bb500440.aspx

[MSDN-DTSTAS] Microsoft Corporation, "DTS Tasks", http://msdn.microsoft.com/en-us/library/aa933506(v=SQL.80).aspx

[MSDN-DTSTRANS] Microsoft Corporation, "DTS Transformations", http://msdn.microsoft.com/en-us/library/aa933491(v=SQL.80).aspx

[MSDN-GetColumnInfo] Microsoft Corporation, "IColumnsInfo::GetColumnInfo", 2008, http://msdn.microsoft.com/en-us/library/ms722704(VS.85).aspx

[MSDN-JscriptRef] Microsoft Corporation, "JScript Language Reference (Windows Scripting - JScript)", http://msdn.microsoft.com/en-us/library/yek4tbz0%28VS.85%29.aspx

[MSDN-POADB] Microsoft Corporation, "Parts of a Database", http://msdn.microsoft.com/en-us/library/aa933066%28SQL.80%29.aspx

[MSDN-PNDT] Microsoft Corporation, "Precision of Numeric Data Types", http://msdn.microsoft.com/en-us/library/ms715867%28v=VS.85%29.aspx

[MSDN-PROPVARIANT] Microsoft Corporation, "PROPVARIANT", http://msdn.microsoft.com/en-us/library/aa380072.aspx

[MSDN-TYPEIND] Microsoft Corporation, "Type Indicators", http://msdn.microsoft.com/en-us/library/ms711251(VS.85).aspx

[PerlScript] ActiveState "ActivePerl 5.8 Documentation", http://docs.activestate.com/activeperl/5.8/Components/Windows/PerlScript.html

1.3 Overview

The Data Transformation Services (DTS) package file format is based on the Compound File Binary file format that is defined in [MS-CFB].

The DTS file format provides for a directory **stream** and several **storages**. Each storage object represents one package. Each package, in turn, has its own directory stream object that can contain multiple storage objects. Each storage represents a single **version** of the package. Each version contains either an encrypted storage (if the **DTS task** or **DTS transformation** supports the **IPersistStorage** interface) or an encrypted stream with the persisted information (either the **IPersistPropertyBag** interface or the default **PropertiesProvider** method) for the version.

1.4 Relationship to Protocols and Other Structures

The DTS package file is stored in a Compound Binary File, as specified in [MS-CFB].

1.5 Applicability Statement

The DTS package structure is always used when persisting a DTS package to a file.

1.6 Versioning and Localization

This document covers versioning issues in the following areas:

- The file format that is described in this document applies to the versions of DTS that are produced by Microsoft® SQL Server® 2000 and Microsoft® SQL Server® 2005. DTS is deprecated in Microsoft® SQL Server® 2008 and Microsoft® SQL Server® 2008 R2 and is replaced by Microsoft® SQL Server® Integration Services (SSIS); however, DTS packages can still be managed and run in SQL Server 2008 and SQL Server 2008 R2. For more information, see [MSDN-DTS] and [MSDN-DTSS2008R2].
- The version **substorage** was originally implemented as a simple substorage of the containing **compound file**, as documented in section 2.2.1. Starting with Microsoft® SQL Server® 7 Service Pack 2, the version substorage was implemented by using an in-memory technique and persisted as a copy of that memory in a substream of the containing compound file, which is named by using the convention that is specified in section 2.2.1.
- Some package variables were added over time. As DTS packages are updated according to the respective versions of Microsoft® SQL Server® for which they were written or updated, the presence or absence of the properties of these DTS packages MUST be taken into account based on the values that are in the version directory entry in the respective version of SQL Server.

1.7 Vendor-Extensible Fields

None.

2 Structures

This section contains the definition of the DTS structure. This document specifically concerns itself with the DTS structure in a package file.

2.1 Compound File

The outer container of the DTS package file format is a compound file. The compound file contains a stream that provides package directory information and multiple substorages, each of which represents a package.

2.1.1 "PackageDirectory" Directory Stream

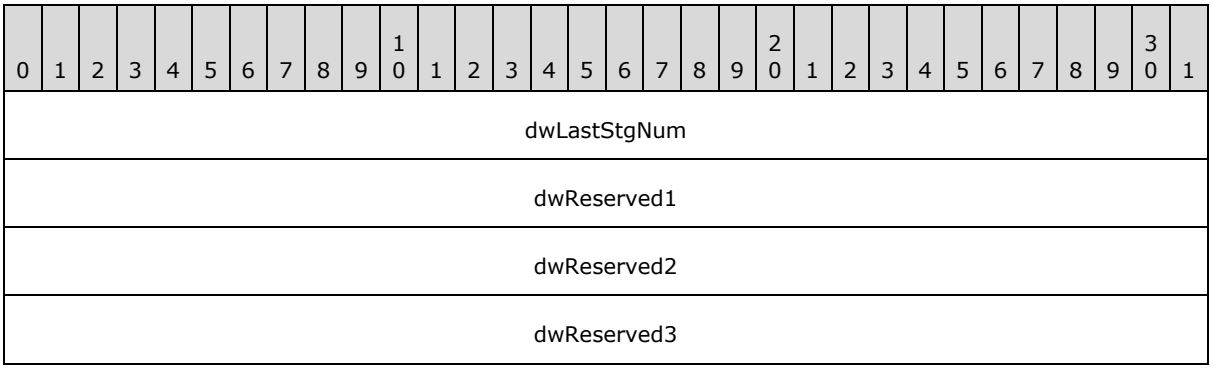
A directory stream named "PackageDirectory" MUST be present in the compound file.

The directory MUST contain a header structure and one or more items. These structures MUST be built in little-endian byte ordering and MUST be reconstructed the same way.

2.1.1.1 "PackageDirectory" Header

The header for the "PackageDirectory" directory stream is stored at the start of the directory stream that is in the compound file.

The header for this directory stream contains the following structure.



dwLastStgNum (4 bytes): A field that contains the last storage number that was used to specify how big the directory is. A newly initialized header would have a value of zero (0).

dwReserved1 (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

dwReserved2 (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

dwReserved3 (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

2.1.1.2 "PackageDirectory" Item

The header for the "PackageDirectory" directory stream is followed by the entries for the individual packages.

The **PackageID** field MUST be unique in the "PackageDirectory" directory stream. The value of **dwStgNum** MUST also be unique and MUST equal the entry number that is in the "PackageDirectory" directory stream.

Each entry contains the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
PackageID																															
...																															
...																															
...																															
szName (WCHAR)																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
(szName (WCHAR) cont'd for 56 rows)																															
CreationDate (DATE)																															
...																															
dwStgNum																															
dwReserved																															

PackageID (16 bytes): A **globally unique identifier (GUID)** that is used to refer to the package.

szName (WCHAR) (256 bytes): A field that specifies the actual name of the stream.

CreationDate (DATE) (8 bytes): A field that specifies the creation date. For more information, see [MS-OAUT], section 2.2.25.

dwStgNum (4 bytes): A field that represents the storage number for the package.

dwReserved (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

2.1.2 Package Storage

Packages are contained in storage subcontainers that are within the compound file. A package is accessed by finding the entry in the "PackageDirectory" directory stream and then opening the storage subcontainer that has a name that consists of the word "Package", plus the storage number, which is expressed as an eight-digit number with leading zeroes.

2.2 Package Structure

Individual packages are persisted in storage containers that are in the compound file. The storage container consists of a version directory with entries for the individual versions of the persisted package and substorages where the version-specific information is persisted.

2.2.1 Package Version Directory

A directory stream named "VersionDirectory" MUST be present in the package container.

This directory MUST contain a header structure and one or more items. These structures MUST be built in little-endian byte ordering and MUST be reconstructed the same way.

2.2.1.1 "VersionDirectory" Header

The header for the "VersionDirectory" directory stream is stored at the start of the directory stream in the compound file.

The header for this directory stream contains the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwLastStgNum																															
LatestVersion (GUID)																															
...																															
...																															
...																															
dwReserved1																															
dwReserved2																															
dwReserved3																															

dwLastStgNum (4 bytes): A newly initialized header would have a value of 0 (zero).

LatestVersion (GUID) (16 bytes): A GUID that is used to refer to the latest version.

dwReserved1 (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

dwReserved2 (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

dwReserved3 (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

2.2.1.2 "VersionDirectory" Item

The header for the "VersionDirectory" directory stream is followed by the items for the individual packages.

The **VersionID** field MUST be unique in the "VersionDirectory" directory stream. The value of the **dwStgNum** variable MUST be unique and MUST equal the entry number that is in the "VersionDirectory" directory stream.

The **dwVersionMajor**, **dwVersionMinor**, and **dwVersionBuild** fields are used to identify any need for special handling requirements that are based on the version of DTS that was used to persist the information about the version of the package. The **dwVersionMajor**, **dwVersionMinor**, and **dwVersionBuild** fields are also used to identify any needs for special handling requirements about the internal structure of all substorages.

Each item contains the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
VersionID (GUID)																															
...																															
...																															
...																															
szDescription (WCHAR [255])																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															

(szDescription (WCHAR [255]) cont'd for 56 rows)
SavedDate (DATE)
...
dwStgNum
dwFlags
dwReserved
dwVersionMajor
dwVersionMinor
dwVersionBuild

VersionID (GUID) (16 bytes): A **GUID** that specifies a unique identifier of the version.

szDescription (WCHAR [255]) (256 bytes): A field that specifies descriptive text for the version.

SavedDate (DATE) (8 bytes): A field that specifies the date that the version was saved. For more information, see [MS-OAUT], section 2.2.25.

dwStgNum (4 bytes): A field that specifies the storage number for the version.

dwFlags (4 bytes): A field that describes the encryption type, if any.

This field can have one or more of the following bits set.

Name	Value	Description
Encrypted	0x00000001	Specifies that the version storage is encrypted.
SimpleCrypt	0x00000002	Specifies that the version storage is encrypted using OLE encryption.
EncryptedDefpwd	0x00000004	Specifies that the default password is set.

dwReserved (4 bytes): A field that MUST be set to zero; nonzero values MUST be ignored.

dwVersionMajor (4 bytes): A field that specifies the major version of the storage file.

dwVersionMinor (4 bytes): A field that specifies the minor version of the storage file.

dwVersionBuild (4 bytes): A field that specifies the build version of the storage file.

2.2.2 Package Version Storage

Package versions are contained in storage subcontainers that are within the package container. A version of a package is accessed by finding the version entry in the **VersionDirectory** directory stream and then opening the storage subcontainer that is in the package container that has a name that consists of the word "Version", plus the storage number, which is expressed as an eight-digit number with leading zeroes.<1>

The data format of the persisted stream consists of a 32-bit length field followed by a format stream [MS-CFB] that represents the substorage or substorages.

2.3 Package Version Structure

Versions of a package are persisted in substorages of the package storage.

The substorage of the package storage contains the following items:

- Persisted password information.
- A stream or substorage, or both, in which the task or information about the transformation is also persisted.

Depending on the credentials and version of the DTS package file format that were used at the time that the persistent storage was created, the persisted information about the transformation can be encrypted.

2.3.1 Encryption

Data for all of the streams within the storage structure, including the directory streams, are encrypted; however, the storage structure itself is not encrypted.<2>

The substorage is encrypted by using an RC2 block encryption algorithm that has a block size of 64 bits. The encryption key that is generated is derived from the hashed bytes of the password of the operator. The length of the key can range from 40 bits to 128 bits, depending on the local machine's encryption configuration.

Every stream in the encrypted substorage has a random **salt** that is stored in the directory for the stream. The salt for the directory stream itself is stored in the directory for the parent storage.

Every salt has a default value of 0x39, 0x67, 0x13, 0xA7, 0x74, 0x8B, 0x0F, 0x77, 0x91, 0xCD, 0x24, 0x8B, 0xB1, 0xAB, 0xEE, 0x32 that is hardcoded in the code for the package. Each root storage directory stream uses this standard salt value.

Data for the stream is encrypted and decrypted 1,000 bytes at a time. Any bytes at the end of the stream are padded at the time of encryption and trimmed at the time of decryption.

The root encrypted storage (the top-level package version storage that is described in section 2.2.1.1) contains the following directory streams:

- Stream "Element00000000" is encrypted with the key that was derived from the hashed bytes of the password of the operator.
- Stream "Element00000001" is encrypted with the key that was derived from the hashed bytes of the password of the owner.

Stream"Element00000001" has the password of the operator stored in it; however, the password of the owner is never stored. All other streams in the version are encrypted with the operator password key.

For security purposes, when the storage is opened, stream "Element00000001" and stream "Element00000000" are also opened to determine whether the user is the owner or the operator.

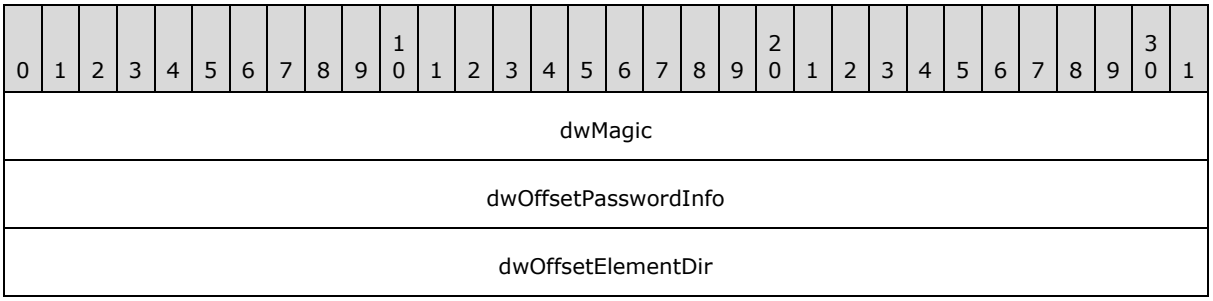
The directory stream contains password information for the storage, and it maps physical stream or storage names that are in the substorage to names that were specified when the element was created and also to the encryption "Salt" for the element.

For example, in a stream named "CollectionProperties", the user would create an entry in the directory stream that is named "CollectionProperties" and would also create the salt value to be used to encrypt the entry. The directory stream itself would be written into an encrypted stream named "Element", plus a storage number (expressed as an eight-digit value with leading zeroes) that would correspond to the entry that is in the directory stream.

In a substorage named "Substorage", the user would create an entry in the directory stream that is named "Substorage" and would also create a salt value. The salt value would be used to encrypt the directory stream of the substorage.

2.3.1.1 Directory Stream Structure

The directory stream starts with the following structure.



dwMagic (4 bytes): A field that contains the number 5577827, which is used to confirm decryption.

dwOffsetPasswordInfo (4 bytes): A field that contains the offset that is in the stream where the **PasswordInfo** structure is located.

dwOffsetElementDir (4 bytes): A field that contains the offset that is in the stream where the "Element" directory is located.

2.3.1.2 PasswordInfo Structure

After the preceding structure, a random number of random bytes are written, and then the **PasswordInfo** structure is created. The **PasswordInfo** structure is 40 bytes in length, although the maximum number of bytes allowed for the password is 32.

The password for the operator is stored in stream "Element00000001". The password that is in stream "Element00000000" is not used; stream "Element00000000" usually contains the string "Fooled you, Oh famous hacker!!" or an arbitrary string.

The **PasswordInfo** structure resembles the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
szOperator Password (WCHAR)																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
(szOperator Password (WCHAR) cont'd for 2 rows)																															

szOperator Password (WCHAR) (40 bytes): A field that specifies the operator password that is used for access.

2.3.1.3 Directory Element

After the **PasswordInfo** structure, a random number of bytes are written, followed by the element entries for the storage.

An entry for the directory element resembles the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
szName																															
...																															
...																															
...																															
...																															
...																															
...																															

...
(szName cont'd for 2 rows)
dwElement
ulSaltLen
Salt (BYTE[16])
...
...
...

szName (40 bytes): A field that specifies the actual name of the stream.

dwElement (4 bytes): A field that specifies the element number to which the name of the stream is being mapped by using the "ElementXXXXXXXX" convention.

ulSaltLen (4 bytes): A field that specifies the length of the salt value for the stream.

Salt (BYTE[16]) (16 bytes): A field that specifies the salt value.

2.4 DTS Object Persistence Structures

Persisted DTS objects save their data by using one of three structures:

- An embedded **PersistStorage** structure.
- A **PropertyBag** structure that is flattened onto a stream.
- A **PropertiesProvider** structure that uses a stream that contains all of the persisted DTS object's Object Linking and Embedding (OLE) Automation properties.

2.4.1 PersistStorage

The **PersistStorage** structure is the default storage mechanism for most DTS objects. **PersistStorage** stores the persistent properties of the object into a substream of the object storage. The substream of the object storage is named "Properties".

PersistStorage also stores any persistent collections into substorages that have the name of the collection. Each substorage, in turn, contains the following items:

- A stream, named "CollectionProperties", that contains a PropertyBag structure that has one variable. This variable is a LONG variable named "Count" that contains the number of collections.
- A substorage named "Item<item number>" for each item in a collection. The substorage contains the properties for the item.

For example, a persisted DTS object that has persistent collections named **Steps**, **Tasks**, **Connections**, and **GlobalVariables** would be persisted by a substorage of the package version

storage. The substorage of the package version storage would contain (at least) a stream named "Properties" and four substorages named "Steps", "Tasks", "Connections", and "GlobalVariables".

2.4.2 PropertiesProvider

The **PropertiesProvider** structure provides a "default" method for custom tasks and transformations that do not implement persistent storage. **PropertiesProvider** uses a stream that contains a PropertyBag structure with the persisted DTS object properties.

This storage method does not support the following types:

- VT_NULL
- VT_EMPTY
- VT_DISPATCH
- VT_UNKNOWN
- VT_ERROR
- VT_ARRAY
- VT_BYREF

2.4.3 PropertyBag

The **PropertyBag** persistent stream is used by both the PersistStorage and PropertiesProvider persistence structures. The stream contains a series of properties that are written as follows.

```
BagStream = 0*BagProp
BagProp = PropName PropSig PropVarType PropValue
; Property name including null terminator
PropName = 1*OCTET %x00
; Signature Bytes
PropSig = %xF7.F7
; Variant Type
PropVarType = UINT16
; Property Value
PropValue = PropObject / PropNull / PropString / PropOther
; PropObject, type is VT_UNKNOWN or VT_DISPATCH
PropObject = ObjSig ObjPersistType ObjID ObjSize ObjData
; PropObject Signature Bytes
ObjSig = %xF3.F3
; PropObject persistence Type
ObjPersistType = PersistTypeStorage / PersistTypeStream
PersistTypeStorage = %x01.00.00.00
PersistTypeStream = %x02.00.00.00
; PropObject ID
ObjID = GUID
; PropObject Size… number of bytes needed to persist the object data
ObjSize = UINT32
; PropObject Data… the bytes of the persisted object data
ObjData = 0*OCTET
; PropNull, type is VT_NULL or VT_EMPTY
PropNull = %x00.00.00.00
; PropString, type is BSTR
PropString = StringSize StringData
```

```
; PropString Size… number of bytes needed to persist the string
StringSize = UINT32
; StringData… bytes of string
StringData = 0*OCTET
; PropOther, any type other than the previous ones. These values are
; converted to strings and persisted that way.
PropOther = OtherSize OtherData
; OtherSize… number of bytes needed to persist the converted string
OtherSize = UINT32
; OtherData … bytes of converted string
OtherData = 0*OCTET
```

2.5 Persisted DTS Objects

2.5.1 Package

The **Package** structure is the highest level object. Its properties and collections are stored by using the PersistStorage structure.

The persistent property set for the **Package** structure contains the following properties.

Property	Type	Description
Name	BSTR	Specifies the name of the **Package** structure.
Description	BSTR	Specifies the description of the **Package** structure.
LogFileName	BSTR	Specifies the full path name of the error log file.
CreatorName	BSTR	Specifies the name of the user who created the package.
CreatorComputerName	BSTR	Specifies the network name of the computer upon which the **Package** structure was created.
PriorityClass	BSTR	Specifies the execution priority for the **Package** structure. For more information, see the **PriorityClass** property table later in this section.
MaxConcurrentSteps	LONG	Specifies the maximum number of steps to execute concurrently on separate threads.
FailOnError	BOOL	Specifies whether the package execution stops if there is an error on a step. If TRUE, the package execution stops. If FALSE, the package execution does not stop.
WriteCompletionStatusToNTEventLog	BOOL	Specifies whether the completion status of the package is written to the application log.<3> If TRUE, the completion status is written to the application log. If FALSE, the completion status is not written to the application log.
LineageOptions<4>	LONG	Specifies how the package execution lineage is

Property	Type	Description
		presented and recorded. For more information, see the **LineageProperty** property table later in this section.
LastExecutionLineage_FullString<5>	BSTR	This lineage field is not used in the persisted file.
LastExecutionLineage_IntegerCRC<6>	LONG	This lineage field is not used in the persisted file.
UseTransaction<7>	BOOL	Specifies whether a transaction is created. If TRUE, a transaction is created. If FALSE, a transaction is not created.
AutoCommitTransaction<8>	BOOL	Specifies whether an active transaction is committed at the end of an execution. If TRUE, an active transaction is committed at the end of an execution. If FALSE, an active transaction is not committed at the end of an execution.
TransactionIsolationLevel<9>	LONG	Specifies the isolation level for a transaction if requested by the **UseTransaction** property when it is set to true. For more information, see the **TransactionIsolationLevel** table later in this section.
RepositoryMetadataOptions<10>	LONG	Specifies metadata scanning and resolution options for the **Package** structure when it is saved to the repository. For more information, see the **RepositoryMetadataOptions** property table later in this section.
UseOLEDBServiceComponents<11>	BOOL	Specifies whether OLE DB service components are used when initializing data sources. If TRUE, OLE DB service components are used. If FALSE, OLE DB service components are not used.
LogToSQLServer<12>	BOOL	Specifies whether a package execution is logged to the database. If TRUE, a package execution is logged to the database. If FALSE, a package execution is not logged to the database.
LogServerName<13>	BSTR	Specifies the name of the server where package logs are written.
LogServerUserName<14>	BSTR	Specifies the user name to be used for login to the log server.
LogServerPassword<15>	BSTR	Specifies the password to be used for login to the log server.
LogServerFlags<16>	LONG	Specifies the flags for the log server.<17> For more information, see the **LogServerFlags**

Property	Type	Description
		property table later in this section.
FailPackageOnLogFailure<18>	BOOL	Specifies whether a package fails if logging fails. If TRUE, the package fails. IF FALSE, the package does not fail.
ExplicitGlobalVariables<19>	BOOL	Specifies whether the **AddGlobalVariable** method is used to add global variables. If TRUE, **AddGlobalVariable** is used to add global variables. If FALSE, **AddGlobalVariable** is not used to add global variables.
PackageType<20>	LONG	Specifies the tool that was used to create the package. For more information, see the **PackageType** property table later in this section.

The **PriorityClass** property has one of the values that are described in the following table.

Name	Value	Description
Low	0x0001	Use Win32 IDLE_PRIORITY_CLASS.
Normal	0x0002	Use Win32 NORMAL_PRIORITY_CLASS.
High	0x0003	Use Win32 HIGH_PRIORITY_CLASS.

The **LineageOptions** property has one or more of the following bits set.

Name	Value	Description
None	0x0000	This bit is not used for file persistence.
AddLineageVariables	0x0001	This bit is not used for file persistence.
WriteToReposIfAvailable	0x0002	This bit is not used for file persistence.
WriteToReposRequired	0x0003	This bit is not used for file persistence.

The **TransactionIsolationLevel** property has one of the values that are described in the following table.

Name	Value	Description
Chaos	0x00000010	Does not hold shared locks and does not implement dirty read protection.
ReadUncommitted (Browse)	0x00000100	Implements dirty read or isolation level 0 locking.
CursorStability (ReadCommitted)	0x00001000	Implements isolation level 1 locking.<21>

Name	Value	Description
RepeatableRead	0x00010000	Implements isolation level 2 locking.
Serializable (Isolated)	0x00100000	Implements isolation level 3 locking

The **RepositoryMetadataOptions** property has one or more of the following bits set. This property is not used for file persistence.

Name	Value	Description
Default	0x0000	Specifies that no scanner resolution is performed.
RequireScannedCatalog	0x0001	Specifies that all database objects are required to be present in the repository.
UseScannedCatalogIfPresent	0x0002	Uses any scanned objects found; nonscanned references create local objects.
ScanCatalogIfNotFound	0x0004	Scans all catalogs that are not already scanned.
ScanCatalogAlways	0x0008	Scans all referenced catalogs, regardless of whether they had already been scanned.

The **LogServerFlags** property can have one of the values that are described in the following table.

Name	Value	Description
Default	0x0000	Uses the default connection method (account/password) to connect to the log server.
UseTrustedConnection	0x0100	Uses integrated security to connect to the log server.<22>

The **PackageType** property can have one of the values that are described in the following table.

Name	Value	Description
Default	0x0000	Default value of the package creator type.
DTSWizard	0x0001	Created by DTS Wizard.
DTSDesigner	0x0002	Created by DTS Query Designer.
SQLReplication	0x0003	Created by SQL Server Replication.
ActiveDirectory	0x0004	Created by Active Directory.

Collections that are persisted as part of the **Package** structure properties contain the following items:

- Steps, persisted in the **Steps** collection. For more information, see Steps Collection.
- Tasks, persisted in the **Tasks** collection. For more information, see DTS Task Objects.

- Connections, persisted in the **Connections** collection. For more information, see Connections Collection.
- Global variables, persisted in the **GlobalVariables** collection. For more information, see GlobalVariables Collection.

Host-specific variables are stored in a substorage named "HostPrivateStorage" by using the **PersistStorage** structure.

2.5.2 Steps Collection

The **Steps** collection is stored as part of the Package object.

The per-item properties of the **Steps** collection are listed in the following table.

Property	Type	Description
Name	BSTR	Specifies the name of the **Step** object.
TaskName	BSTR	Specifies the name of the task to execute.
ActiveXScript	BSTR	Specifies the ActiveX script to execute prior to the task.
ScriptLanguage	BSTR	Specifies the ActiveX script language (such as Visual Basic Scripting Edition (VBScript), Jscript, or PerlScript). For more information about VBScript, see [MSDN-VBSLR]. For more information about Jscript, see [MSDN-JscriptRef]. For more information about PerlScript, see [PerlScript].
CommitSuccess	BOOL	Specifies whether to commit the **Step** if it completes successfully. If TRUE, the **Step** is committed. If FALSE, the **Step** is not committed.
RollbackFailure	BOOL	Specifies whether to roll back the package transaction if the **Step** fails. If TRUE, the package transaction is rolled back if the **Step** fails. If FALSE, the package transaction is not rolled back if the **Step** fails.
CloseConnection	BOOL	Specifies whether to close the connection when the **Step** completes. If TRUE, the connection is closed. If FALSE, the connection is not closed.
RelativePriority	LONG	Specifies the relative priority of the thread where the **Step** runs. For more information, see the **RelativePriority** property table later in this section.
AddGlobalVariables	BOOL	Specifies whether global variables can be accessed. If TRUE, global variables can be accessed. If FALSE, global variables cannot be accessed.
ExecuteInMainThread	BOOL	Specifies whether to use the main thread or a worker thread. If TRUE, the main thread is used. If FALSE, the worker thread is used.
IsPackageDSORowset	BOOL	Specifies whether the **Step** returns a rowset (if the containing

Property	Type	Description
		package is a rowset provider). If TRUE, the **Step** returns a rowset. If FALSE, the **Step** does not return a rowset.
JoinTransactionIfPresent	BOOL	Specifies whether the **Step** executes within the package transaction. If TRUE, the **Step** executes within the package transaction. If FALSE, the **Step** does not execute within the package transaction.
IsDisabled	BOOL	Specifies whether the **Step** is to be presently disabled. If TRUE, the **Step** is disabled. If FALSE, the **Step** is not disabled.
FailPackageOnError	BOOL	Specifies whether the package fails if the **Step** commits an error. If TRUE, the package fails. If FALSE, the package does not fail.

The **RelativePriority** property can have one of values that are described in the following table.

Name	Value	Description
Lowest	0x0001	Use Win32 THREAD_PRIORITY_LOWEST.
BelowNormal	0x0002	Use Win32 THREAD_PRIORITY_BELOW_NORMAL.
Normal	0x0003	Use Win32 THREAD_PRIORITY_NORMAL.
AboveNormal	0x0004	Use Win32 THREAD_PRIORITY_ABOVE_NORMAL.
Highest	0x0005	Use Win32 THREAD_PRIORITY_HIGHEST.

The persistent collection set contains precedence constraints, persisted in a PersistStorage structure that is named "PrecedenceConstraints". Each row of the persistent collection can have the values that are described in the following table.

Property	Type	Description
StepName	BSTR	Specifies the name of the **Step**, the result of which should be evaluated to determine whether the **DTS constraint** is satisfied.
PrecedenceBasis	LONG	Specifies how to determine whether the DTS constraint is satisfied. For more information, see the **PrecedenceBasis** property table.
Value	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the value of the object.

The **PrecedenceBasis** property can have one of the values that are described in the following table.

Name	Value	Description
Execute Status	0x0000	Used for comparison with the current **Step** status.
Execute Result	0x0001	Used for comparison with the completed **Step** result.

The value of the execution status (used for comparison) can be one of the values that are described in the following table.

Name	Value	Description
Waiting	0x0001	Specifies that the **Step** is waiting to execute.
In Progress	0x0002	Specifies that the **Step** is executing.
Inactive	0x0003	Specifies that the **Step** is not active.
Completed	0x0004	Specifies that the execution of the **Step** is complete.

The value of the execution results (used for comparison) can be one of the values that are described in the following table.

Name	Value	Description
Success	0x0000	Specifies that the **Step** succeeded.
Failure	0x0001	Specifies that the **Step** failed.
Retry	0x0002	Not applicable.

2.5.3 DTS Task Objects

The task class ID is stored in a PropertyBag structure named "TaskID" in a property named "TaskID".

If the task supports a **PropertyBag**, it is stored in a stream named "TaskPropertyBag"; otherwise, its properties collection is persisted in a stream named "TaskPropertiesCollection".

2.5.3.1 Internal Objects

2.5.3.1.1 ActiveX Script

The class ID of the **ActiveX Script** task is (0x10020907, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The **ActiveX Script** task object uses the PersistStorage structure to store its properties and collections.

The persistent property set for the **ActiveX Script** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
FunctionName	BSTR	Specifies the name of the function entry point.

Property	Type	Description
ScriptLanguage	BSTR	Specifies the script language (such as "VBScript" or "PerlScript").
ActiveXScript	BSTR	Specifies the ActiveX source script.
AddGlobalVariables	BOOL	Specifies whether the GlobalVariables collection is added to the namespace. If TRUE, the **GlobalVariables** collection is added to the namespace. If FALSE, the **GlobalVariables** collection is not added to the namespace.
Description	BSTR	Not applicable.

2.5.3.1.2 DataPump

The **DataPump** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **DataPump** task is (0x10020908, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **DataPump** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
SourceObjectName	BSTR	Used only for the **IOpenRowset** OLE DB object (such as "Customer").
DestinationObjectName	BSTR	Used to open a simple rowset on the destination if no SQL statement is specified (such as "Sales").
ExceptionFileName	BSTR	Specifies the **UNC** file name to write exceptions to. If null, exceptions are not logged to file and the DTS **DataPump** task object fails on first error.
SourceSQLStatement	BSTR	Specifies that the SQL statement creates a source rowset from the command object.
DestinationSQLStatement	BSTR	Specifies that the SQL statement is to create a destination rowset from the command object.
SourceConnectionID	LONG	Specifies the source connection.
DestinationConnectionID	LONG	Specifies the destination connection.
ProgressRowCount	LONG	Specifies the frequency of notifications that are sent to the connection point by the DTS **DataPump** task object.
MaximumErrorCount	LONG	Specifies the maximum number of errors that are allowed before the DTS **DataPump** task object aborts.
FetchBufferSize	LONG	Specifies the number of rows to fetch from the OLE DB source.
UseFastLoad	BOOL	Specifies the use of FastLoad for inserts.

Property	Type	Description
InsertCommitSize	LONG	Specifies the number of rows to insert in a single transaction when FastLoad is being used.
ExceptionFileColumnDelimiter	BSTR	Specifies the column delimiter that is in the exception file.
ExceptionFileRowDelimiter	BSTR	Specifies the row delimiter for the data that is in the exception file.
AllowIdentityInserts	BOOL	Specifies whether the SET IDENTITY_INSERT Transact-SQL statement is set to ON before execution and OFF after execution by the DTS **DataPump** task object.
FirstRow	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the first row to be used in the operation that the DTS **DataPump** task object performs.
LastRow	VARIANT	Specifies the last row to be used in the operation.
FastLoadOptions	LONG	Specifies the bitmask of FastLoad options. For more information, see the **FastLoadOptions** value table later in this section.
ExceptionFileOptions	LONG	Specifies the bitmask of exception file options. For more information, see the **ExceptionFileOptions** value table later in this section.
ExceptionFileTextQualifier	BSTR	Specifies the text qualifier for the data in the exception file.
InputGlobalVariableNames	BSTR	Specifies a list of global variables that are to be used as query parameters.
DataPumpOptions	LONG	Specifies the bitmask of the DTS **DataPump** task options. For more information, see the **DataPumpOptions** value table later in this section.

The **FastLoadOptions** value can have one or more of the following bits set.

Option	Value
Keep nulls	0x0001
Check DTS constraints	0x0002
Lock table	0x0004

The **ExceptionFileOptions** value can have one or more of the following bits set.

Option	Value
Log all data to a single file	0x00000001
Log errors to file	0x00000002

Option	Value
Log source row of error to file	0x00000004
Log destination row of error to file	0x00000008
Write ANSI file	0x00000100
Check DTS constraints	0x00000200
Lock table	0x00000400
Overwrite existing file (0=append)	0x00001000
Abort pump if data logging fails	0x00002000

The **DataPumpOptions** value can have the following bit set.

Option	Value
Always commit final batch	0x00000001

The persistent collection set for the DTS **DataPump** task contains the following items:

- Transformation properties, persisted in a container that is named "Transformations". For information about the data that is required for each row of the persistent collection set, see Transformation Object.
- Source command properties, persisted in a container that is named "SourceCommandProperties". Each row of the persistent collection set can have the following values.

Property	Type	Description
SetGroup	GUID	Specifies the property set or property group GUID.
ID	LONG	Specifies the property ID.
Value	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the property value.

- Destination command properties, persisted in a container that is named "DestinationCommandProperties". Each row of the persistent collection set can have the following values.

Property	Type	Description
SetGroup	GUID	Specifies the property set or property group GUID.
ID	LONG	Specifies the property ID.
Value	VARIANT	Specifies the property value.

- Destination column definitions, persisted in a container that is named "DestinationColumnDefinitions". For more information, see Column Collection.
- Lookups, persisted in a container that is named "Lookups". For more information, see Lookup Collection.

2.5.3.1.3 Execute Package

The **Execute Package** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Execute Package** task object is (0x10020912, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Execute Package** task object contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
ServerName	BSTR	Specifies the name of the server upon which to execute.
ServerUserName	BSTR	Specifies the user name of the account to use on the server.
ServerPassword	BSTR	Specifies the password of the account to use on the server.
RepositoryDatabaseName	BSTR	Specifies the name of the repository database (if required).
PackageName	BSTR	Specifies the name of the package to execute.
PackagePassword	BSTR	Specifies that a password is needed to access the package.
PackageID	BSTR	Specifies the GUID of the package.
VersionID	BSTR	Specifies the GUID of the package version to use.
FileName	BSTR	Specifies the file name of the package.
UseTrustedConnection	BOOL	Specifies whether to use a trusted connection.<23>
UseRepository	BOOL	Specifies whether to use the repository to access a package. If TRUE, the repository is used to access a package. If FALSE, the repository is not used to access a package.
InputGlobalVariableNames	BSTR	Specifies a list of global variables that are used as parameters.

2.5.3.1.4 Parallel Data Pump

The class ID of the **Parallel Data Pump** task is (0x10020911, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The **Parallel Data Pump** task object uses the PersistStorage structure to store its properties and collections.

The persistent property set for the **Parallel Data Pump** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.

Property	Type	Description
SourceObjectName	BSTR	Used only for **IOpenRowset** (such as "Customer").
DestinationObjectName	BSTR	Used to open a simple rowset upon the destination if no SQL statement is specified (such as "Sales").
ExceptionFileName	BSTR	Specifies the UNC file name to write exceptions to. If null, exceptions are not logged to file and the pump fails on first error.
SourceSQLStatement	BSTR	Specifies the SQL statement to use to create a source rowset from a command object.
DestinationSQLStatement	BSTR	Specifies the SQL statement to use to create a destination rowset from a command object.
SourceConnectionID	LONG	Specifies the source connection.
DestinationConnectionID	LONG	Specifies the destination connection.
TransformationSetOptions	LONG	Specifies the bitmask for use with the **TransformationSetOptions** value. For more information, see the following **TransformationSetOptions** value table.
InputGlobalVariableNames	BSTR	Specifies a list of global variables that are used as query parameters.

The **TransformationSetOptions** value can have one or more of the following bits set.

Option	Value
Hierarchical (0=flattened)	0x0001
Data Driven Queries	0x0004

The persistent collection set contains the following items:

- Transformation sets, persisted in a container that is named "TransformationSets". For more information, see DTS Transformation Sets.
- Source command properties, persisted in a container that is named "SourceCommandProperties". Each row of the persistent collection set can have the following values.

Property	Type	Description
SetGroup	GUID	Specifies the property set or property group GUID.
ID	LONG	Specifies the property ID.
Value	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the property value.

- Destination command properties, persisted in a container that is named "DestinationCommandProperties". Each row of the persistent collection set can have the following values.

Property	Type	Description
SetGroup	GUID	Specifies the property set or property group GUID.
ID	LONG	Specifies the property ID.
Value	VARIANT	Specifies the property value.

2.5.3.1.5 Data Driven Query

The **Data Driven Query** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Data Driven Query** task is (0x1002090D, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Data Driven Query** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of task or lineage.
SourceObjectName	BSTR	Used only for **IOpenRowset** (such as "Customer").
DestinationObjectName	BSTR	Used to open a simple rowset upon destination if no SQL statement is specified (such as "Sales").
ExceptionFileName	BSTR	Specifies the UNC file name to write exceptions to. If null, exceptions are not logged to file, and the pump fails on first error.
SourceSQLStatement	BSTR	Specifies the SQL statement to use to create a source rowset from the command object.
DestinationSQLStatement	BSTR	Specifies the SQL statement to use to create a destination rowset from the command object.
SourceConnectionID	LONG	Specifies the source connection.
DestinationConnectionID	LONG	Specifies the destination connection.
ProgressRowCount	LONG	Specifies the frequency of notifications that are sent to the connection point by the DTS Data Pump task.
MaximumErrorCount	LONG	Specifies the maximum number of errors that are allowed before the DTS Data Pump task aborts.
FetchBufferSize	LONG	Specifies the number of rows to fetch from the OLE DB source.
InsertQuery	BSTR	Specifies the query to use when a value of Insert is specified.
UpdateQuery	BSTR	Specifies the query to use when a value of Update is specified.

Property	Type	Description
DeleteQuery	BSTR	Specifies the query to use when a value of Delete is specified.
UserQuery	BSTR	Specifies the query to use when a value of User query is specified.
ExceptionFileColumnDelimiter	BSTR	Specifies the column delimiter for the data that is in the exception file.
ExceptionFileRowDelimiter	BSTR	Specifies the row delimiter for the data that is in the exception file.
FirstRow	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the first row to be used in the operation.
LastRow	VARIANT	Specifies the last row to be used in the operation.
ExceptionFileOptions	LONG	Specifies the bitmask for use with the **ExceptionFileOptions** value. For more information, see the **ExceptionFileOptions** value table later in this section.
ExceptionFileTextQualifier	BSTR	Specifies the text qualifier for the data that is in the exception file.
InputGlobalVariableNames	BSTR	Specifies a list of global variables that are to be used as query parameters.

The **ExceptionFileOptions** value can have one or more of the following bits set.

Option	Value
Log all data to a single file	0x00000001
Log errors to file	0x00000002
Log source row of error to file	0x00000004
Log destination row of error to file	0x00000008
Write ANSI file	0x00000100
Check DTS constraints	0x00000200
Lock table	0x00000400
Overwrite existing file (0=append)	0x00001000
Abort pump if data logging fails	0x00002000

2.5.3.1.6 Create Process

The **Create Process** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Create Process** task is (0x10020909, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Create Process** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
CommandLine	BSTR	Specifies the Win32 command line.
SuccessReturnCode	LONG	Specifies the value of the return code that specifies success.
Timeout	LONG	Specifies the number of seconds before the task aborts. A value of 0 indicates that there is no timeout.
FailPackageOnTimeout	BOOL	Specifies that the task is terminated if timeout occurs. If TRUE, the task is terminated. If FALSE, the task is not terminated.

2.5.3.1.7 Execute SQL

The **Execute SQL** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Execute SQL** task is (0x1002090C, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Execute SQL** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
SQL	BSTR	Specifies the SQL query/command string.
ConnectionID	LONG	Specifies the connection to use.
InputGlobalVariableNames	BSTR	Specifies the list of global variables that are to be used as query parameters.
OutputGlobalVariableNames	BSTR	Specifies the list of global variables that can be populated by the SQL command.
Output as **Recordset**	BOOL	Specifies whether output is in the form of a recordset. If TRUE, output is in the form of a recordset. If FALSE, output is not in the form of a recordset.

The persistent collection set contains command properties, persisted as "Command Properties".

Each row of the persistent collection set can have the following values.

Property	Type	Description
SetGroup	GUID	Specifies the property set or property group GUID.
ID	LONG	Specifies the property ID.
Value	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the property value.

2.5.3.1.8 Transfer Database Objects

The **Transfer Database Objects** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Transfer Database Objects** task object is (0x1002090E, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Transfer Database Objects** task object contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
SourceServer	BSTR	Specifies the server to use for the source.
SourceLogin	BSTR	Specifies the login to use against the source server.
SourcePassword	BSTR	Specifies the password to use against the source server.
SourceDB	BSTR	Specifies the database to use in the source server.
DestinationServer	BSTR	Specifies the server to use for the destination.
DestinationLogin	BSTR	Specifies the login to use on the destination server.
DestinationPassword	BSTR	Specifies the password to use on the destination server.
DestinationDB	BSTR	Specifies the database to use on the destination server.
ScriptDirectory	BSTR	Specifies the location where generated scripts should be saved.
SourceTrustedLogin	BOOL	Specifies whether to use a trusted connection.<24>
DestinationTrustedLogin	BOOL	Specifies whether to use a trusted connection.<25>
IncludeDependencies	BOOL	Specifies whether to include all dependent objects. If TRUE, all dependent objects are included. If FALSE, all dependent objects are not included.
IncludeLogin	BOOL	Specifies whether to include logins from the source. If TRUE, logins from source are included. If FALSE, logins from source are not included.
IncludeUsers	BOOL	Specifies whether to include users and roles from the source. If TRUE, users and roles from the source are included.

Property	Type	Description
		If FALSE, users and roles from the source are not included.
DropDestObjects	BOOL	Specifies whether to drop the destination object first. If TRUE, the destination object is dropped first. If FALSE, the destination object is not dropped first.
CopySchema	BOOL	Specifies whether to transfer schema from the source to the destination. If TRUE, schema is transferred from source to destination. If FALSE, schema is not transferred from source to destination.
CopyDataOption	LONG	Specifies a bitmask for use with the **CopyDataOption** value. For more information, see the **CopyDataOption** value table later in this section.
ScriptOption	LONG	Specifies scripting options. For more information, see the **ScriptOption** value table later in this section.
ScriptOptionEx	LONG	Specifies extended scripting options. For more information, see the **ScriptOptionEx** value table later in this section.
CopyAllObjects	BOOL	Specifies whether to copy all objects. If TRUE, all objects are copied. If FALSE, all objects are not copied.
SourceTranslateChar	BOOL	Specifies whether to translate from the source character set. If TRUE, the source character set is translated. If FALSE, the source character set is not translated.
DestTranslateChar	BOOL	Specifies whether to translate to the destination character set. If TRUE, the destination character set is translated. If FALSE, the destination character set is not set.
DestUseTransaction	BOOL	Specifies a destination operation in one transaction. If TRUE, a destination operation is specified. If FALSE, a destination operation is not specified.
UseCollation	BOOL	Specifies the use destination’s collation.

The **CopyDataOption** value can have one or more of the following bits set.

Option	Value
ReplaceData	0x0001
AppendData	0x0002

The **ScriptOption** value can have one or more of the following bits set.

Option	Value
Include object drops.	0x00000001
Include object permissions.	0x00000002
Include object creation.	0x00000004
Include clustered index creation.	0x00000008
Include script triggers.	0x00000010
Include script database permissions.	0x00000020
Script to file only.	0x00000040
Include rule/default bindings.	0x00000100
Do not include declarative referential integrity (DRI). For more information about DRI, see [MSDN-POADB].	0x00000200
Convert user-defined types to base type.	0x00000400
Include "if not exists".	0x00001000
Include nonclustered index creation.	0x00002000
Script aliases.	0x00004000
Do not append "go" to commands.	0x00008000
Script DRI indexes as indexes for noDRI.	0x00010000
Include descriptive headers in output.	0x00020000
Owner-qualify DROP [CREATE] statements.	0x00040000
Convert timestamp columns to binary.	0x00080000
Append sorted_data for clusters.	0x00100000
Append sorted_data and fillfactor for clusters.	0x00200000

The **ScriptOptionEx** value can have one or more of the following bits set.

Option	Value
SET ANSI PADDING on/off before CREATE TABLE.	0x00000001
Generate ANSI output file.	0x00000002
Generate UNICODE output file.	0x00000004
Ignore errors in script file generation.	0x00000008
Do not generate 'ON <fg>' for replication.	0x00000010
Mark system triggers (replication only).	0x00000020
Only script user triggers (replication only).	0x00000040

Option	Value
Script encrypted password.	0x00000080
Script XP to a separate file.	0x00000100
Do not script what-if indexes.	0x00000200
Script notification for SQL Agent.	0x00000400
Include job in alert scripting.	0x00000800
Include full-text index scripting.	0x00080000
Include the security identifier (SID) for standard logins.<26>	0x00100000
Include full-text catalog scripting.	0x00200000
Include extended property scripting.	0x00400000
Do not script collation clauses.<27>	0x00800000
Do not use certain features in the output script.<28>	0x01000000
Disable job at end of job creation.	0x02000000

2.5.3.1.9 Send Mail

The **Send Mail** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Send Mail** task is (0x1002090F, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Send Mail** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
Profile	BSTR	Specifies the Messaging Application Programming Interface (MAPI) profile to use to send email.
Password	BSTR	Specifies the password for the MAPI profile.
ToLine	BSTR	Specifies the list of To message recipients.
CCline	BSTR	Specifies the list of Cc message recipients.
FileAttachments	BSTR	Specifies the list of files to attach to an email message.
Subject	BSTR	Specifies the subject line of an email message.
Message	BSTR	Specifies the body text of an email message.
IsNTService	BOOL	Specifies that email is sent from an operating system service.<29>
SaveMailInSentItemsFolder	BOOL	Specifies that a copy of a sent email message is saved in the "Sent

Property	Type	Description
		Items" folder. If TRUE, a copy of a sent email message is saved in the "Sent Items" folder. If FALSE, a copy of a sent email message is not saved in the "Sent Items" folder.

2.5.3.1.10 Bulk Insert

The **Bulk Insert** task object uses the PersistStorage structure to store its properties and collections.

The class ID of the **Bulk Insert** task is (0x10020910, 0xEB1C, 0x11CF, 0xAE, 0x6E, 0x0, 0xAA, 0x0, 0x4A, 0x34, 0xD5).

The persistent property set for the **Bulk Insert** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
Codepage	BSTR	Specifies the code page hint (such as "ACP", "OEM", "RAW", or code page number).
FieldTerminator	BSTR	Specifies the character that separates fields, such as a tab ("\t") (this is the default).
RowTerminator	BSTR	Specifies the character that separates rows, such as newline ("\n") (this is the default).
DestinationTableName	BSTR	Specifies the name of the table to bulk copy into.
DataFile	BSTR	Specifies the full path of the data file with data to be copied.
FormatFile	BSTR	Specifies the full path of the associated format file.
SortedData	BSTR	Specifies the column to ORDER BY. The default is FALSE, meaning that no column is specified.
CheckConstraints	BOOL	Specifies that DTS constraints are checked before inserting data. If TRUE, DTS constraints are checked before inserting data. If FALSE, DTS constraints are not checked before inserting data.
KeepIdentity	BOOL	Specifies that identity values are kept when importing data. If TRUE, identity values are kept when importing data. If FALSE, identity values are not kept when importing data.
KeepNulls	BOOL	Specifies whether nulls are kept or filled with defaults. If TRUE, nulls are kept. If FALSE, nulls are filled with defaults.
TableLock	BOOL	Specifies whether a table is locked for insert. IF TRUE, the table is locked for insert. If FALSE, the table is not locked for insert.

Property	Type	Description
ConnectionID	LONG	Specifies the connection to use.
BatchSize	LONG	Specifies the maximum number of rows that are in each batch.
FirstRow	LONG	Specifies the first row from the input rowset to copy.
LastRow	LONG	Specifies the last row from the input rowset to copy.
MaximumErrors	LONG	Specifies the maximum number of errors that can occur before the **Bulk Insert** operation fails.
DataFileType	LONG	Specifies the type of data in the input file. For more information, see the **DataFileType** property table later in this section.

The **DataFileType** property can have one of the following values:

Property	Type	Description
char	0x0000	Copy from a data file that contains character data.
native	0x0001	Copy from a data file that contains native database types. **Note** Native database types are created by the bcp utility. For more information about the bcp utility, see [MSDN-BCPU].
widechar	0x0002	Copy from a data file containing UNICODE character data.
widenative	0x0003	Same as **native**, but **char**, **varchar**, and text columns are stored as UNICODE in the bcp-generated file.

2.5.3.2 External Objects

2.5.3.2.1 DTS Message Queue Task

The class ID for the **DTS Message Queue** task is (4C237FFF-FC07-11D2-ACF5-00C04F689068). The programmatic identifier (ProgID) for the **DTS Message Queue** task is "DTSMessageQueueTask".

Variables for the **DTS Message Queue** task are persisted in a substream named "DTSMessageQueueTask". The stream consists of the following.

```
MQTask = MQSig MQName MQDesc MQDirection MQPath MQMsgType MQDataFile MQPackage MQComparison
MQTimeout MQRemove MQMessages
; Signature. 32-bit little-endian "1" to signify this is a persistence version.
MQSig = %x01.00.00.00
; Name
MQName = 0*OCTET %x00
; Description
MQDesc = 0*OCTET %x00
; Direction… Sender or Receiver
MQDirection = MQSender / MQReceiver
MQSender = %x00.00.00.00
MQReceiver = %x01.00.00.00
```

```
; Queue Path
MQPath = 0*OCTET %x00
; Receive Message Type (GlobalVariables, String or DataFile)
MQMsgType = MQString / MQDataFile / MQGlobalVariables
MQString = %x00.00.00.00
MQDataFile = %x01.00.00.00
MQGlobalVariables = %x02.00.00.00
; Data File information
MQDataFile = DataFileName DataFileNoOverWrite
DataFileName = 0*OCTET %x00
DataFileNoOverWrite = INT32
; Package information
MQPackage = PackageID VersionID LineageID
PackageID = 0*OCTET %x00
VersionID = 0*OCTET %x00
LineageID = 0*OCTET %x00
MQRemove MQMessages
; Comparison
MQComparison = CompareType CompareValue
; Type of comparison.
CompareType = CompareNone / CompareExact / CompareIgnoreCase / CompareContains
CompareNone = %x00.00.00.00
CompareExact = %x01.00.00.00
CompareIgnoreCase = %x02.00.00.00
CompareContains = %x03.00.00.00
; Value to compare
CompareValue= 0*OCTET %x00
; Receive Message Timeout value
MQTimeout = INT32
; Should messages be removed from queue?
MQRemove = INT32
; Messages, in a counted list.
MQMessages = MessageCount 0*Message
MessageCount = INT32
; Message
Message = MQSig MessageType MessageWait MessageTransaction MessageValue
; Message Type (GlobalVariables, String or DataFile)
MessageType = MessageString / MessageDataFile / MessageGlobalVariables
MessageString = %x00.00.00.00
MessageDataFile = %x01.00.00.00
MessageGlobalVariables = %x02.00.00.00
; Should we wait for ACK?
MessageWait = INT32
; Should we use a transaction?
MessageTransaction = INT32
; Value
MessageValue = MsgStringValue / MsgDataFileName / MsgGlobalVarName
MsgStringValue = 0*OCTET %x00
MsgDataFileName = 0*OCTET %x00
MsgGlobalVarName = 0*OCTET %x00
```

2.5.3.2.2 DTS FTP Task

The class ID for the **DTS FTP** task is 0E090A39-E613-11D2-ACED-00C04F689068. The ProgID for the **DTS FTP** task is "DTSFTPTask".

The **DTS FTP** task uses the same mechanism as the PersistStorage structure to store its persistent properties, except that the stream that is used to persist the properties of the **DTS FTP** task is named "Contents".

The persistent property set for the **DTS FTP** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.
Description	BSTR	Specifies the description of the task or lineage.
DestSite	BSTR	Specifies the destination directory for operation of the task.
SourceFilename	BSTR	Specifies the list of files to transfer by using the task.
SourcePassword	BSTR	Specifies the password to be used to connect to source.
SourceUsername	BSTR	Specifies the user name to be used to connect to the source.
SourceSite	BSTR	Specifies the location from which files will be transferred.
SourceLocation	LONG	Specifies the kind of source location. For more information, see the **SourceLocation** property table later in this section.
NonOverwriteable	BOOL	Specifies whether the task should fail if the file that is to be transferred already exists in destination directory. If TRUE, the task fails. If FALSE, the task does not fail.
NumRetriesOnSource	LONG	Specifies the number of retries before the task fails.

The **SourceLocation** property can have one of the following values

Name	Value	Description
InternetSite	0	Specifies that the source location is an FTP server that is on the Internet.
Directory	1	Specifies that the source location is a directory.

2.5.3.2.3 DTS Bulk Load From XML Task

The class ID for the **DTS Bulk Load From XML** task is (68D98B6F-5E6A-11D4-B53F-00C04FA120B1). The ProgID for the **DTS Bulk Load From XML** task is "DTSTask.DTSSQLXMLBulkLoadTask".

The **DTS Bulk Load From XML** task uses the same mechanism as the PersistStorage structure to store its persistent properties, except that the stream that is used to persist the properties of the **DTS Bulk Load From XML** task is named "Contents".

The persistent property set for the **DTS Bulk Load From XML** task contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the task.

Property	Type	Description
Description	BSTR	Specifies the description of the task or lineage.
ConnectionMethod	LONG	Specifies the connection method. For more information, see the following table for the **ConnectionMethod** property.
ServerName	BSTR	Specifies the server name. For more information, see the following table for the **ServerName** property.
DatabaseName	BSTR	Specifies the database name. For more information, see the following table for the **DatabaseName** property.
UseTrustedConnection	BOOL	Specifies whether to use a trusted connection.<30>
UserName	BSTR	Specifies the user ID or user name to use when making a connection.
Password	BSTR	Specifies the password to use when making a connection.
ConnectionString	BSTR	Used where a connection string is specified in **ConnectionMethod**.
ConnectionID	LONG	Specifies the ID of the connection to use where a connection object is specified in **ConnectionMethod**.
KeepIdentity	BOOL	Specifies whether to keep identity values when importing data. If TRUE, identity values are kept when importing data. If FALSE, identity values are not kept when importing data.
KeepNulls	BOOL	Specifies whether nulls should be kept or filled with defaults. If TRUE, nulls are kept. If FALSE, nulls are filled with defaults.
CheckConstraints	BOOL	Specifies whether data constraints are kept before inserting data. If TRUE, data constraints are kept before inserting data. If FALSE, data constraints are not kept before inserting data.
ForceTableLock	BOOL	Specifies whether the table should be locked for insert. If TRUE, the table is locked for insert. If FALSE, the table is not locked for insert.
PerformBulkLoad	BOOL	Specifies whether a bulk load should be performed.
XMLFragment	BOOL	Specifies whether the source data is an XML fragment.
PerformSchemaGeneration	BOOL	Specifies whether to create the required tables before performing a bulk load.
DropCreateTables	BOOL	Specifies whether already-existing tables should be dropped and re-created.
UseID	BOOL	Specifies whether the ID type attribute in the mapping schema can be used in creating a primary key constraint when the table is created.
Options	LONG	This bit is not used.
UseTransaction<31>	BOOL	Specifies whether the bulk load should be done as a transaction,

Property	Type	Description
		which will roll back if the load fails.
ErrorLogFile	BSTR	Specifies the file name into which errors and messages should be logged. If the string is empty, no logging takes place.
SchemaFile	BSTR	Specifies the path to an XSD schema file.
DataStream	BSTR	Specifies the path to an XML data stream.
DataSourceFormat	LONG	Specifies the format of the source data. For more information, see the following table for the **DataSourceFormat** property.
BulkLoadEngine	LONG	Specifies the bulk load engine. For more information, see the following table for the **BulkLoadEngine** property.

The **ConnectionMethod** property can have one of the values in the following table.

Name	Value	Description
Parameters	0x00000100	Specifies to use connection parameters.
String	0x00000001	Specifies to use a connection string.
DTSObject	0x00000002	Specifies to use a DTS connection object.

The **BulkLoadEngine** property can have one of the values in the following table.

Type	Value	Description
SQLXMLObject	0x00000001	Specifies to use the **SQLXMLBulkLoad** task object.
SQLOLEDB	0x00000002	Specifies to use the **XML Bulk Load** extension of OLE DB.

The **DataSourceFormat** property can have one of the values in the following table.

Type	Value	Description
Text	0x00000001	Specifies that the data source is text.
Stream	0x00000002	Specifies that the data source is a stream.

2.5.3.3 Dynamic Properties

The class ID for the **Dynamic Properties** task is (CF1988D3-E143-11D2-AB67-00C04F79EE8C). The ProgID for the **Dynamic Properties** task is "DTSDynamicPropertiesTask".

Variables for the **Dynamic Properties** task are persisted in a substream named "DynamicPropertiesTask". The substream consists of the following.

```
DynPropTask = DynPropSig DynPropName DynPropDesc DynPropList
; Signature = 1 to signify this is a persistence version
DynPropSig = %x01.00.00.00
; Name of task
```

```
DynPropName = 0*OCTET %x00
; Description of task
DynPropDesc = 0*OCTET %x00
; Property List
DynPropList = DynPropCount 0*DynProp
; Property Count
DynPropCount = INT32
; Property
DynProp = DynPropSig DynPropID DynPropType DynPropValue
; Property ID
DynPropID = 0*OCTET %x00
; Property Type
DynPropType = TypeIniFile / TypeQuery / TypeGlobalVariable / TypeEnvironmentVariable /
TypeConstant / TypeDataFile
TypeIniFile = %x00.00.00.00
TypeQuery = %x01.00.00.00
TypeGlobalVariable = %x02.00.00.00
TypeEnvironmentVariable = %x03.00.00.00
TypeConstant = %x04.00.00.00
TypeDataFile = %x05.00.00.00
; Property Value
DynPropValue = DynPropInifile / DynPropQuery / DynPropGlobalVar / DynPropConstant /
DynPropDataFile / DynPropEnvironment
; Inifile Value
DynPropInifile = InifileName InifileSection InifileKey
InifileName = 0*OCTET %x00
InifileSection = 0*OCTET %x00
InifileKey = 0*OCTET %x00
; Query Value
DynPropQuery = QueryConnectionID QuerySQL
QueryConnectionID = UINT32
QuerySQL = 0*OCTET %x00
; Global Variable Value
DynPropGlobalVar = GlobalVarName
GlobalVarName = 0*OCTET %x00
; Constant
DynPropConstant = ConstantValue
ConstantValue = 0*OCTET %x00
; Data File
DynPropDataFile = DataFileName
DataFileName = 0*OCTET %x00
; Environment Variable
DynPropEnvironment = EnvironmentVariableName
EnvironmentVariableName = 0*OCTET %x00
```

2.5.4 DTS Transformation Sets

The **TransformationSet** object uses the PersistStorage structure to store its properties and collections.

The persistent property set for the **TransformationSet** object contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the **TransformationSet**.
Description	BSTR	Specifies the description of the **TransformationSet**.

Property	Type	Description
ExceptionFileName	BSTR	Specifies the UNC file name to write exceptions to. If null, exceptions are not logged to the file and the DTS DataDump task fails on first error.
ProgressRowCount	LONG	Specifies the frequency of notifications sent to the connection point by the DTS DataPump task.
MaximumErrorCount	LONG	Specifies the maximum number of errors allowed before the DTS DataPump task aborts.
SourceSQLStatement	BSTR	Specifies the SQL statement used to create the source rowset from the command object.
FetchBufferSize	LONG	Specifies the number of rows to fetch from the OLE DB source.
ExceptionFileColumnDelimiter	BSTR	Specifies the column delimiter that is in the exception file.
ExceptionFileRowDelimiter	BSTR	Specifies the row delimiter for the data that is in the exception file.
FirstRow	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the first row to be used in the data transformation operation.
LastRow	VARIANT	Specifies the last row to be used in the operation.
InsertQuery	BSTR	Specifies the query to use when a value of Insert is specified.
UpdateQuery	BSTR	Specifies the query to use when a value of Update is specified.
DeleteQuery	BSTR	Specifies the query to use when a value of Delete is specified.
UserQuery	BSTR	Specifies the query to use when a value of User query is specified.
ExceptionFileOptions	LONG	Specifies the bitmask for exception file options. For more information, see the following table for the **ExceptionFileOptions** property.
ExceptionFileTextQualifier	BSTR	Specifies the text qualifier for the data that is in the exception file.

The **ExceptionFileOptions** property can have one or more of the following bits set.

Option	Value
Log all data to a single file	0x00000001
Log errors to file	0x00000002
Log source row of error to file	0x00000004

Option	Value
Log destination row of error to file	0x00000008
Write ANSI file	0x00000100
Check data constraints	0x00000200
Lock table	0x00000400
Overwrite existing file (0=append)	0x00001000
Abort transformation if data logging fails	0x00002000

The persistent collection set contains the following items:

- Transformations, persisted in a container that is named "Transformations". For more information, see 2.5.4.1.
- Lookups, persisted in a container that is named "Lookups". For more information, see section 2.5.6.
- Column definitions, persisted in a container that is named "ColumnDefinitions". For more information, see section 2.5.7.
- Insert query columns, persisted in a container that is named "InsertQueryColumns". For more information, see section 2.5.7.
- Update query columns, persisted in a container that is named "UpdateQueryColumns". For more information, see section 2.5.7.
- Delete query columns, persisted in a container that is named "DeleteQueryColumns". For more information, see section 2.5.7.
- User query columns, persisted in a container that is named "UserQueryColumns". For more information, see section 2.5.7.

2.5.4.1 Transformation Object

The **Transformation** object uses the PersistStorage structure to store its properties and collections.

The persistent property set for the **Transformation** object contains the following items.

Property	Type	Description
Name	BSTR	Specifies the name of the transformation.
ServerID	BSTR	Specifies the ProgID or class ID of the server object.
ServerParameters	VARIANT	Specifies the initialization parameter for the server object.
TransformFlags	LONG	Specifies the flags that indicate characteristics. For more information, see the following table for the **TransformFlags** property.
ForceSourceBlobsBuffered	LONG	Specifies the buffer source BLOBs. For more information, see the following table for the **ForceSourceBlobsBuffered** property.

Property	Type	Description
ForceBlobsInMemory	BOOL	Specifies whether to use memory allocation for BLOBs. If TRUE, memory allocation is used for BLOBs. If FALSE, memory allocation is not used for BLOBs.
InMemoryBlobSize	LONG	Specifies the byte size of per-column allocation of memory for BLOBs.
TransformPhases	LONG	Specifies the phases that this transformation supports. For more information, see the following table for the **TransformPhases** property.

The **ForceSourceBlobsBuffered** property can have one of the values in the following table.

Name	Value	Description
Default	0x00000000	Server uses its default buffering logic for source BLOBs.
Always	0x00000001	Server always buffers source BLOBs.
Never	0x00000002	Server never buffers source BLOBs.

The **TransformFlags** property can have one or more of the following bits set.

Name	Value	Description
AllowDemotion	0x00000001	Allows the transfer to proceed even if there are potential overflows.
AllowPromotion	0x00000002	Allows the transfer to proceed when there is promotion in the data range.
AllowStringTruncation	0x00000004	Allows column (w)char or byte data to be truncated silently.
AllowNumericTruncation	0x00000008	Allows the transfer to proceed even when numeric truncation is possible.
AllowNullChange	0x00000010	Allows the transfer to proceed even if the source column allows NULL values and the destination column does not.
AllowSignChange	0x00000020	Allows the transfer to proceed even in the event that the source and destination have a signed versus unsigned mismatch.
RequireExactType	0x00000040	Requires that the data type of the destination column be exactly the same as the data type of the source column.
ForceConvert	0x00000080	Allows the conversion to proceed at all times, even when the source and destination types are fundamentally different.
PreserveDestRows	0x00000100	Causes the data pump to not clear the destination row storage at the end of row processing.
AllowLosslessConversion	0x00000200	Allows all conversions for which a lossless conversion is possible.

The **TransformPhases** property can have one of the values in the following table.

Name	Value	Description
PreSourceData	0x00000001	Occurs before the first source row is processed.
PostSourceData	0x00000002	Occurs after all source rows are processed.
Transform	0x00000004	Occurs after the source row is fetched; performs the primary transformation processing.
OnTransformFailure	0x00000008	Occurs after transformation fails (such as a conversion error).
OnInsertSuccess	0x00000010	Occurs after an Insert operation or a data-driven query succeeds.
OnInsertFailure	0x00000020	Occurs after an Insert operation or a data driven query fails.
OnBatchComplete	0x00000040	Occurs after a fast load batch completes, whether a success or failure.
OnPumpComplete	0x00000080	Occurs once at end of a DTS data pump operation.

The persistent collection for the **Transformation** object contains the following items:

- Source columns, persisted in a container that is named "SourceColumns". For more information, see Column Collection.
- Destination columns, persisted in a container that is named "DestinationColumns". For more information, see Column Collection.

2.5.4.1.1 DateTime String

The **DateTime String** transformation persists its variables by using a PropertyBag structure on a stream that is named "DateTimeStringProperties".

The class ID for the **DateTime String** transformation is (0x10010b01, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

The variables for the **DateTime String** transformation are persisted to the **PropertyBag** structure in the following order.

Property	Type	Description
Month1LongName	BSTR	Specifies the long name of month 1 (that is, January).
Month2LongName	BSTR	Specifies the long name of month 2 (that is, February).
Month3LongName	BSTR	Specifies the long name of month 3 (that is, March).
Month4LongName	BSTR	Specifies the long name of month 4 (that is, April).
Month5LongName	BSTR	Specifies the long name of month 5 (that is, May).
Month6LongName	BSTR	Specifies the long name of month 6 (that is, June).
Month7LongName	BSTR	Specifies the long name of month 7 (that is, July).
Month8LongName	BSTR	Specifies the long name of month 8 (that is, August).

Property	Type	Description
Month9LongName	BSTR	Specifies the long name of month 9 (that is, September).
Month10LongName	BSTR	Specifies the long name of month 10 (that is, October).
Month11LongName	BSTR	Specifies the long name of month 11 (that is, November).
Month12LongName	BSTR	Specifies the long name of month 12 (that is, December).
Month1ShortName	BSTR	Specifies the short name of month 1 (that is, Jan).
Month2ShortName	BSTR	Specifies the short name of month 2 (that is, Feb).
Month3ShortName	BSTR	Specifies the short name of month 3 (that is, Mar).
Month4ShortName	BSTR	Specifies the short name of month 4 (that is, Apr).
Month5ShortName	BSTR	Specifies the short name of month 5 (that is, May).
Month6ShortName	BSTR	Specifies the short name of month 6 (that is, Jun).
Month7ShortName	BSTR	Specifies the short name of month 7 (that is, Jul).
Month8ShortName	BSTR	Specifies the short name of month 8 (that is, Aug).
Month9ShortName	BSTR	Specifies the short name of month 9 (that is, Sep).
Month10ShortName	BSTR	Specifies the short name of month 10 (that is, Oct).
Month11ShortName	BSTR	Specifies the short name of month 11 (that is, Nov).
Month12ShortName	BSTR	Specifies the short name of month 12 (that is, Dec).
Day1LongName	BSTR	Specifies the long name of day 1 (that is, Monday).
Day2LongName	BSTR	Specifies the long name of day 2 (that is, Tuesday).
Day3LongName	BSTR	Specifies the long name of day 3 (that is, Wednesday).
Day4LongName	BSTR	Specifies the long name of day 4 (that is, Thursday).
Day5LongName	BSTR	Specifies the long name of day 5 (that is, Friday).
Day6LongName	BSTR	Specifies the long name of day 6 (that is, Saturday).
Day7LongName	BSTR	Specifies the long name of day 7 (that is, Sunday).
Day1ShortName	BSTR	Specifies the short name of day 1 (that is, Mon).
Day2ShortName	BSTR	Specifies the short name of day 2 (that is, Tue).
Day3ShortName	BSTR	Specifies the short name of day 3 (that is, Wed).
Day4ShortName	BSTR	Specifies the short name of day 4 (that is, Thu).
Day5ShortName	BSTR	Specifies the short name of day 5 (that is, Fri).
Day6ShortName	BSTR	Specifies the short name of day 6 (that is, Sat).
Day7ShortName	BSTR	Specifies the short name of day 7 (that is, Sun).

Property	Type	Description
InputFormat	BSTR	Specifies format of the **datetime** value that is in the source.
OutputFormat	BSTR	Specifies format of the **datetime** value that is in the destination.
AMSymbol	BSTR	Specifies the suffix for times before 12:00 noon.
PMSymbol	BSTR	Specifies the suffix for times on and after 12:00 noon.
ShortYear2000Cutoff	LONG	Specifies the 2-digit year below which the year is 20xx.

2.5.4.1.2 Uppercase String

The **Uppercase String** transformation has no internal variables, and therefore it has no persisted data.

The class ID for the **Uppercase String** transformation is (0x10010701, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

2.5.4.1.3 Lowercase String

The **Lowercase String** transformation has no internal variables, and therefore has no persisted data.

The class ID for the **Lowercase String** transformation is (0x10010801, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

2.5.4.1.4 Middle of String

The **Middle of String** transformation persists its variables by using a PropertyBag structure on a stream that is named "DataPumpMidStringProperties".

The class ID for the **Middle of String** transformation is (0x10010901, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

The variables for the **Middle of String** transformation are persisted to the **PropertyBag** structure in the following order.

Property	Type	Description
CharacterStart	LONG	Specifies the index of the starting character.
CharacterCount	LONG	Specifies the count of characters, including the beginning character.
TrimLeadingWhiteSpace	BOOL	Specifies whether trim should lead white space characters. If TRUE, trim leads white space characters. If FALSE, trim does not lead white space characters.
TrimTrailingWhiteSpace	BOOL	Specifies whether trim should trail white space characters. If TRUE, trim trails white space characters. If FALSE, trim does not trail white space characters.
TrimEmbeddedWhiteSpace	BOOL	Specifies whether trim should be embedded in white space characters.

Property	Type	Description
		IF TRUE, trim is embedded in white space characters. If FALSE, trim is not embedded in white space characters.
UpperCaseString	BOOL	Specifies whether to fold the string to uppercase. If TRUE, the string is folded to uppercase. If FALSE, the string is not folded to uppercase.
LowerCaseString	BOOL	Specifies whether to fold the string to lowercase. If TRUE, the string is folded to lowercase. If FALSE, the string is not folded to lowercase.

2.5.4.1.5 Trim String

The **Trim String** transformation persists its variables by using a PropertyBag structure on a stream that is named "DataPumpTrimStringProperties".

The class ID for the **Trim String** transformation is (0x10010d01, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

The variables for the **Trim String** transformation are persisted to the **PropertyBag** structure in the following order.

Property	Type	Description
TrimLeadingWhiteSpace	BOOL	Specifies whether trim should lead white space characters. If TRUE, trim leads white space characters. If FALSE, trim does not lead white space characters.
TrimTrailingWhiteSpace	BOOL	Specifies whether trim should trail white space characters. If TRUE, trim trails white space characters. If FALSE, trim does not trail white space characters.
TrimEmbeddedWhiteSpace	BOOL	Specifies whether trim should be embedded in white space characters. If TRUE, trim is embedded in white space characters. If FALSE, trim is not embedded in white space characters.
UpperCaseString	BOOL	Specifies whether to fold the string to uppercase. If TRUE, the string is folded to uppercase. If FALSE, the string is not folded to uppercase.
LowerCaseString	BOOL	Specifies whether to fold the string to lowercase. If TRUE, the string is folded to lowercase. If FALSE, the string is not folded to lowercase.

2.5.4.1.6 Read File

The **Read File** transformation persists its variables by using a PropertyBag structure on a stream that is named "DataPumpReadFileProperties".

The class ID for the **Read File** transformation is (0x10010a01, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

The variables for the **Read File** transformation are persisted to the **PropertyBag** structure in the following order.

Property	Type	Description
ErrorIfFileNotFound	BOOL	Specifies whether an error should be raised if a file is not found. If TRUE, an error is raised if a file is not found. If FALSE, an error is not raised if a file is not found.
FilePath	BSTR	Specifies the directory to append at the beginning of the file name as a prefix.
UnicodeFile	BOOL	Specifies whether data is to be read in Unicode. If TRUE, data is read in Unicode. If FALSE, data is not read in Unicode.
OEMFile	BOOL	Specifies whether data is to be read in the OEM character set. If TRUE, data is read in the OEM character set. If FALSE, data is not read in the OEM character set.

2.5.4.1.7 Write File

The **Write File** transformation persists its variables by using a PropertyBag structure on a stream that is named "DataPumpWriteFileProperties".

The class ID for the **Write File** transformation is (0x10010c01, 0x740b, 0x11d0, 0xae, 0x7b, 0x0, 0xaa, 0x0, 0x4a, 0x34, 0xd5).

The variables for the **Write File** transformation are persisted to the **PropertyBag** structure in the following order.

Property	Type	Description
ErrorIfFileExists	BOOL	Specifies whether an error should be raised if a file already exists. If TRUE, an error is raised if a file already exists. If FALSE, an error is not raised if a file already exists.
AppendIfFileExists	BOOL	Specifies whether write should append to an already-existing file. If TRUE, the already-existing file is appended. If FALSE, the already-existing file is replaced.
FilePath	BSTR	Specifies the path to the prefix of the file name.
FileColumnName	BSTR	Specifies the name of the column that contains the name of the file that is to be written.
UnicodeFile	BOOL	Specifies whether data is to be read in Unicode. If TRUE, data is read in Unicode. If FALSE, data is not read in Unicode.
OEMFile	BOOL	Specifies whether data is to be read in OEM character set.

Property	Type	Description
		If TRUE, data is read in OEM character set. If FALSE, data is not read in OEM character set.

2.5.5 GlobalVariables Collection

The **GlobalVariables** collection is stored as part of the Package object.

The per-item properties of the **GlobalVariables** collection are listed in the following table.

Property	Type	Description
Name	BSTR	Specifies the name of the global variable.
Value	VARIANT ([MS-OAUT] section 2.2.29)	Specifies the value of the global variable.

2.5.6 Lookup Collection

The **Lookup** collection is used in multiple objects.

The per-item properties of the **Lookup** collection are listed in the following table.

Property	Type	Description
Name	BSTR	Specifies the name of the lookup.
Query	BSTR	Specifies the name of the SQL query.
ConnectionID	LONG	Specifies the connection to use for the SQL query.
MaxCacheRows	LONG	Specifies the maximum number of rows to cache when fetching from the OLE DB source.

2.5.7 Column Collection

The **Column** collection is used in multiple objects. This protocol references commonly used data types as defined in [MS-DTYP].

The per-item properties of the **Column** collection are listed in the following table.

Property	Type	Description
Name	BSTR	Specifies the name of the column.
Ordinal	LONG	Specifies the ordinal of the column.
Flags	LONG	A bitmask that describes column characteristics. See the following table for bits in the **Flags** field.
Size	LONG	The maximum possible length of a value in the column. For columns that use a fixed-length data type, this is the size of the data type. For columns that use a variable-length data type, this is one of the following: ▪ The maximum length of the column in characters (for DBTYPE_STR and DBTYPE_WSTR) or in bytes (for DBTYPE_BYTES and

Property	Type	Description
		DBTYPE_VARNUMERIC), if one is defined. For example, a CHAR(5) column in an SQL table has a maximum length of 5. ▪ The maximum length of the data type in characters (for DBTYPE_STR and DBTYPE_WSTR) or in bytes (for DBTYPE_BYTES and DBTYPE_VARNUMERIC) if the column does not have a defined length. ▪ ~0 (bitwise, the value is not 0; all bits are set to 1) if neither the column nor the data type has a defined maximum length. For data types that do not have a length, this is set to ~0 (bitwise, the value is not 0; all bits are set to 1).
DataType	LONG	The indicator of the column's data type. If the data type of the column varies from row to row, this must be DBTYPE_VARIANT. For a list of valid type indicators, see [MSDN-TYPEIND].
Precision	LONG	If **DataType** is a numeric data type, this is the D precision of the column. The precision of columns with a data type of DBTYPE_VARNUMERIC, DBTYPE_DECIMAL, or DBTYPE_NUMERIC depends on the definition of the column. For the precision of all other numeric data types, see [MSDN-PNDT]. For DBTYPE_DBTIMESTAMP data types, this is the length of the string representation (assuming the maximum allowed precision of the fractional seconds component). If the column's data type is not **numeric** or **datetime**, this is ~0 (bitwise, the value is not 0; all bits are set to 1).
Scale	LONG	If **DataType** is DBTYPE_DECIMAL or DBTYPE_NUMERIC, this is the number of digits to the right of the decimal point. Otherwise, this is ~0 (bitwise, the value is not 0; all bits are set to 1). For DBTYPE_DBTIMESTAMP data types, this is the length of the string representation of the fractional seconds component.
Nullable	BOOL	Specifies whether the column can contain a NULL.
ColumnID	VARIANT ([MS-OAUT] section 2.2.29)	The column ID of the column. For more information, see [MSDN-COLID].

The **Flags** property can have one or more of the following bits set. For more information, see [MSDN-GetColumnInfo].

Name	Value	Description
ISBOOKMARK	0x00000001	Set if the column contains a bookmark.
MAYDEFER	0x00000002	Set if the column is deferred, meaning that data is not fetched by the provider until the data is explicitly requested.
WRITE	0x00000004	Set if writes are permitted to the column.
WRITEUNKNOWN	0x00000008	Set if permission to write to the column is not known.
ISFIXEDLENGTH	0x00000010	Set if all data in the column has the same length.

Name	Value	Description
ISNULLABLE	0x00000020	Set if the column can be set to NULL or if it is unknown whether the column can be set to NULL.
MAYBENULL	0x00000040	Set if column data can include NULL values or if it is unknown whether column data can include NULL values.
ISLONG	0x00000080	Set if the column contains very long data.
ISROWID	0x00000100	Set if the column contains a persistent, unique, read-only value that is used for row identification.
ISROWVER	0x00000200	Set if the column contains a persistent, read-only value that is used for row-level version identification.
CACHEDEFERRED	0x00001000	Set if the data for the column is currently in a local cache.
ISCHAPTER	0x00002000	Set if the column contains a chapter value.

The **DataType** property can have one of the following values. For more information, see [MSDN-TYPEIND].

Name	Value	Description
EMPTY	0	No type specified for the value.
NULL	1	A NULL value.
I2	2	A two-byte, signed integer. For more information, see **SHORT**.
I4	3	A four-byte, signed integer. For more information, see **LONG**.
R4	4	A single-precision, floating point value. For more information, see **FLOAT**.
R8	5	A double-precision, floating point value. For more information, see **DOUBLE**.
CY	6	A currency value, stored in an 8-byte signed integer. For more information, see CURRENCY ([MS-OAUT] section 2.2.24).
DATE	7	A date value, stored in a double-precision floating point format. For more information, see DATE ([MS-OAUT] section 2.2.25).
BSTR	8	A counted null-terminated string value. For more information, see **BSTR**.
IDISPATCH	9	A pointer to an **IDispatch** interface. For more information, see [MS-OAUT] section 3.1.
ERROR	10	A four-byte error code. For more information, see SCODE ([MS-OAUT] section 2.2.48).
BOOL	11	A Boolean value, stored in a 2-byte integer. For more information, see VARIANT_BOOL ([MS-OAUT] section 2.2.27).
VARIANT	12	A container that can hold many types of data. For more information, see VARIANT ([MS-OAUT] section 2.2.29.2).
IUNKNOWN	13	A pointer to an object.

Name	Value	Description
DECIMAL	14	A representation of a decimal number using an eight-byte structure. For more information, see DECIMAL ([MS-OAUT] section 2.2.26).
I1	16	A one-byte, signed integer. For more information, see **CHAR**.
UI1	17	A one-byte, unsigned integer. For more information, see **BYTE**.
UI2	18	A two-byte, unsigned integer. For more information, see **USHORT**.
UI4	19	A four-byte, unsigned integer. For more information, see **ULONG**.
I8	20	An eight-byte, signed integer. For more information, see **LARGE_INTEGER**.
UI8	21	An eight-byte, unsigned integer. For more information, see **ULARGE_INTEGER**.
FILETIME	64	A time value stored in a 64-bit structure. For more information, see **FILETIME**.
GUID	72	A globally-unique identifier. For more information, see **GUID**.
BYTES	128	An array of BYTEs. For more information, see **BYTE**.
STR	129	A null-terminated character string. For more information, see **CHAR**.
WSTR	130	A null-terminated UNICODE character string. For more information, see **wchar_t**.
NUMERIC	131	A 19-byte structure representing a variable-scale, signed number. For more information, see [MSDN-TYPEIND].
UDT	132	A user-defined type. For more information, see [MSDN-TYPEIND].
DBDATE	133	A six-byte date structure. For more information, see [MSDN-TYPEIND].
DBTIME	134	A six-byte time structure. For more information, see [MSDN-TYPEIND].
DBTIMESTAMP	135	A 16-byte date/time structure accurate to billionths of a second. For more information, see [MSDN-TYPEIND].
HCHAPTER	136	A four-byte chapter value that can identify rows in a child rowset.
PROPVARIANT	138	A structure that can hold many types of data. For more information, see [MSDN-PROPVARIANT].
VARNUMERIC	139	A variable length structure to represent a precise decimal number. For more information, see [MSKB229884].
VECTOR	0x1000	A bitmask that is added to another type to indicate that the column contains a DBVECTOR structure that refers to the value. For more information, see [MSDN-TYPEIND].
RESERVED	0x8000	Reserved.

2.5.8 Connections Collection

The **Connections** collection is stored as part of the Package object.

The per-item properties of the **Connections** collection are listed in the following table.

Property	Type	Description
Name	BSTR	Specifies the name of the connection.
ID	LONG	Specifies the connection ID.
Reusable	BOOL	Specifies whether the connection is reusable. If TRUE, the connection is reusable. If FALSE, the connection is not reusable.
ConnectImmediate	BOOL	Specifies whether an immediate connection is made. If TRUE, an immediate connection is made. If FALSE, an immediate connection is not made.
DataSource	BSTR	Specifies the name of the data source.
UserID	BSTR	Specifies the user ID or user name to use when making a connection.
Password	BSTR	Specifies the password to use when making a connection.
ConnectionTimeout	LONG	Specifies the number of seconds to wait for a connection.
ProviderID	BSTR	Specifies the program ID of the OLE DB provider.
Catalog	BSTR	Specifies the name of the catalog for a connection.
UseTrustedConnection	BOOL	Specifies whether to use a trusted connection.<32>
Description	BSTR	Specifies the connection description.
UseDSL	BOOL	Specifies whether to use a Data Source Link (DSL) file. If TRUE, DSL is used. If FALSE, DSL is not used.
UDLPath	BSTR	Specifies the full path to the DSL file to be used to create a connection.

The persistent collection contains connection properties, persisted as "Connection Properties". "Connection Properties" contains provider-specific OLE DB connection properties.

3 Structure Examples

3.1 Compound File Example

This section provides an example of a compound file.

A compound file named "DTS Sample" might contain the following information:

- A directory stream named "PackageDirectory" that has the following contents.

```
0000-0010:  00 00 00 00-00 00 00 00-3c 3b 0a 01-01 00 00 00  ........ <;......
0000-0020:  76 3e 9c 6e-d4 a6 5e 4a-90 8f f2 5f-e3 f0 e2 d5  v>.n..^J ..._....
0000-0030:  44 00 54 00-53 00 20 00-53 00 61 00-6d 00 70 00  D.T.S... S.a.m.p.
0000-0040:  6c 00 65 00-00 00 00 00-00 00 00 00-00 00 00 00  l.e..... ........
0000-0050:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ........ ........
0000-0060:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ........ ........
0000-0070:  00 00 00 00-00 00 00 00-00 39 0a 01-10 5c 0a 01  ........ .9...\..
0000-0080:  00 00 00 00-00 00 00 00-e8 93 06 00-2d 07 a7 77  ........ ....-..w
0000-0090:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ........ ........
0000-00a0:  d0 3d 00 00-a4 94 06 00-33 f8 a6 77-9c 94 06 00  .=...... 3..w....
0000-00b0:  00 00 00 00-c8 5b 0a 01-d8 53 0a 01-00 00 00 00  .....[.. .S......
0000-00c0:  c8 5c 0a 01-58 3d b3 77-00 00 00 00-00 00 00 00  .\..X=.w ........
0000-00d0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ........ ........
0000-00e0:  00 00 00 00-00 00 00 00-5b 3b f8 77-90 f1 fc 77  ........ [;.w...w
0000-00f0:  37 3b f8 77-d4 17 0a 01-00 00 00 00-e4 17 0a 01  7;.w.... ........
0000-0100:  b8 5f 0a 01-a0 94 06 00-10 f3 0d 00-a0 93 06 00  ._...... ........
0000-0110:  50 82 12 00-04 a0 06 00-a7 9d fb 77-78 3c f8 77  P....... ...wx<.w
0000-0120:  ff ff ff ff-7c 94 06 00-0c 3c f8 77-00 00 00 00  ....|... .<.w....
0000-0130:  bc 17 0a 01-58 3d b3 77-b4 94 06 00-0b 2f f8 77  ....X=.w ...../.w
0000-0140:  d4 17 0a 01-00 00 00 00-a6 3e e8 77-d4 17 0a 01  ........ .>.w....
0000-0150:  b8 17 0a 01-55 ef a6 77-67 ef a6 77-f0 94 06 00  ....U..w g..w....
0000-0160:  d8 c1 a6 77-32 b1 a6 77-00 00 a8 0a-00 00 00 00  ...w2..w ........
0000-0170:  d4 94 06 00-ea c1 a6 77-00 00 00 00-c8 5c 0a 01  .......w .....\..
0000-0180:  a4 f4 a6 77-b8 5f 0a 01-7c 95 06 00-c8 5c 0a 01  ...w._.. |....\..
0000-0190:  00 01 03 80-c8 5c 0a 01-c8 5c 0a 01-00 00 00 00  .....\.. .\......
0000-01a0:  00 00 00 00-10 5c 0a 01-01 00 00 00-4c 95 06 00  .....\.. ....L...
0000-01b0:  4c 95 06 00-76 bc a6 77-00 00 00 00-00 00 00 00  L...v..w ........
0000-01c0:  00 00 00 00-7c 95 06 00-71 75 a7 77-58 3d b3 77  ....|... qu.wX=.w
0000-01d0:  c2 03 00 00-00 00 00 00-00 00 00 00-3c d4 e9 77  ........ ....<..w
0000-01e0:  a8 4e e8 77-ff ff ff ff-60 95 06 00-58 3d b3 77  .N.w.... `...X=.w
0000-01f0:  02 00 03 80-f4 9f 06 00-01 00 00 00-00 00 00 00  ........ ........
0000-0200:  00 00 00 00-58 3d b3 77-84 95 06 00-44 a2 a9 77  ....X=.w ....D..w
0000-0210:  c4 9d 06 00-c2 03 00 00-04 00 00 00-00 00 00 00  ........ ........
0000-0220:  7c 95 06 00-00 00 00 00-00 00 00 00-12 ab 06 00  |....... ........
0000-0230:  fe 36 c5 e8-16 3d e2 40-00 00 00 00-00 00 00 00  .6...=.@ ........
```

- A storage named "Package00000000" that contains the following streams:
  - A stream named "VersionDirectory".
  - A stream named "Version00000000".

3.2 Package Container Example

The following figure illustrates the structure of a typical DTS container file.

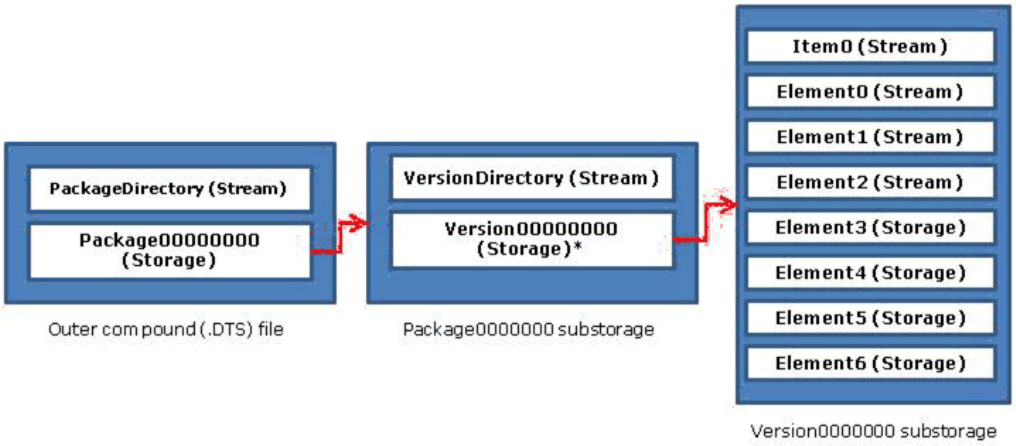


Figure 1: Structure of a DTS container file

The items that are contained in the Version0000000 substorage column are specified in the following table.

Item	Description
Item0 (Stream)	Contains either the default password stream or, if passwords are used, a red-herring stream.
Element0 (Stream)	Directory stream, encrypted by using the operator password.
Element1 (Stream)	Directory stream, encrypted by using the owner password.
Element2 (Stream)	Persistent property stream for the data object.
Element3 (Stream)	Steps collection storage.
Element4 (Stream)	**Tasks** collection storage.
Element5 (Stream)	Connections collection storage.
Element6 (Stream)	GlobalVariables collection storage.

Version0000000 substorage can be persisted as either a storage or as a stream of an in-memory storage structure.

4 Security

4.1 Security Considerations for Implementers

None.

5 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® SQL Server® 7
- Microsoft® SQL Server® 7.0.533
- Microsoft® SQL Server® 7.0.115
- Microsoft® SQL Server® 7.5
- Microsoft® SQL Server® 2000
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2012

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.2: The storage subcontainer is implemented as a substorage in versions of the DTS package file format that are prior to SQL Server 7.0.533 (using the **dwVersionMajor**, **dwVersionMinor**, and **dwVersionBuild** fields in the version directory). The subcontainer is persisted using a substream that is named according to the same convention that was used for substorages in prior versions of SQL Server.

<2> Section 2.3.1: In the Microsoft implementation, substorages of a package are encrypted by using Windows CryptoAPI version 1.0.

<3> Section 2.5.1: In the Microsoft implementation, the application log is the Windows "Application" log.

<4> Section 2.5.1: This property was added in SQL Server 7.0.533.

<5> Section 2.5.1: This property was added in SQL Server 7.0.533.

<6> Section 2.5.1: This property was added in SQL Server 7.0.533.

<7> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.544.

<8> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.544.

<9> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.544.

<10> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.544.

<11> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.544.

<12> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<13> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<14> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<15> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<16> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<17> Section 2.5.1: Only defined values specify that Windows Authentication is used to log in to the log server.

<18> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<19> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<20> Section 2.5.1: This property was added in Microsoft SQL Server 7.0.80.

<21> Section 2.5.1: This option is the default for SQL Server.

<22> Section 2.5.1: In the Microsoft implementation, this is Windows NT Integrated Security.

<23> Section 2.5.3.1.3: In the Microsoft implementation, the **UseTrustedConnection** property specifies whether to use Windows Authentication security mode.

If TRUE, Windows Authentication security mode is used.

If FALSE, Windows Authentication security mode is not used.

<24> Section 2.5.3.1.8: In the Microsoft implementation, the **SourceTrustedLogin** property specifies whether to use Windows Authentication for connection to the source.

If TRUE, Windows Authentication is used for connection to the source.

If FALSE, Windows Authentication is not used for connection to the source.

<25> Section 2.5.3.1.8: In the Microsoft implementation, the **DestinationTrustedLogin** property specifies whether to use Windows Authentication for connection to the destination.

If TRUE, Windows Authentication is used for connection to the destination.

If FALSE, Windows Connection is used for connection to the destination.

<26> Section 2.5.3.1.8: In the Microsoft implementation, this value indicates to include the SID for standard logins to SQL Server.

<27> Section 2.5.3.1.8: In the Microsoft implementation, this value indicates that script collation clauses are not used from a SQL Server 7.x source.

<28> Section 2.5.3.1.8: In the Microsoft implementation, this value indicates that features in SQL Server 7 are not used.

<29> Section 2.5.3.1.9: In the Microsoft implementation, the **IsNTService** property indicates whether a Windows NT Service is used.

If TRUE, email is sent from a Windows NT Service.

If FALSE, email is not sent from a Windows NT Service.

<30> Section 2.5.3.2.3: The **UseTrustedConnection** property specifies whether to use Windows Authentication.

If TRUE, Windows Authentication is used.

If FALSE, Windows Authentication is not used.

<31> Section 2.5.3.2.3: This property was added in SQL Server 7.0.554.

<32> Section 2.5.8: In the Microsoft implementation, the **UseTrustedConnection** property specifies whether to use Windows Authentication security mode.

If TRUE, Windows Authentication is used.

If FALSE, Windows Authentication is not used.

6 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

7 Index