Session Class

The Session object controls the installation process. It opens the install database, which contains the installation tables and data.

Methods

Method	Description
Dispose(disposing)	Closes the session handle. Also closes the active database handle, if it is open. After closing a handle, further method calls may throw an «see T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException» .
DoAction(action)	Executes a built-in action, custom action, or user-interface wizard action.
DoAction(action, actionData)	Executes a built-in action, custom action, or user-interface wizard action.
DoActionSequence(sequenceTable)	Executes an action sequence described in the specified table.
EvaluateCondition(condition)	Evaluates a logical expression containing symbols and values.
EvaluateCondition(condition, defaultValue)	Evaluates a logical expression containing symbols and values, specifying a default value to be returned in case the condition is empty.

Method	Description
Format(format)	Formats a string containing installer properties.
FormatRecord(record)	Returns a formatted string from record data.
FormatRecord(record, format)	Returns a formatted string from record data using a specified format.
FromHandle(handle, ownsHandle)	Creates a new Session object from an integer session handle.
GetMode(mode)	Gets the designated mode flag for the current install session.
GetProductProperty()	Retrieves product properties (not session properties) from the product database.
GetSourcePath()	Gets the full path to the designated folder on the source media or server image.
GetTargetPath()	Gets the full path to the designated folder on the installation target drive.
GetTotalCost()	Gets the total disk space per drive required for the installation.
Log(msg)	Writes a message to the log, if logging is enabled.
Log(format, args)	Writes a formatted message to the log, if logging is enabled.
Message(messageType, record)	Performs any enabled logging operations and defers execution to the UI handler object associated with the engine.

Method	Description
SetInstallLevel(installLevel)	Sets the install level for the current installation to a specified value and recalculates the Select and Installed states for all features in the Feature table. Also sets the Action state of each component in the Component table based on the new level.
SetMode(mode, value)	Sets the designated mode flag for the current install session.
SetTargetPath()	Sets the full path to the designated folder on the installation target drive.
VerifyDiskSpace()	Checks to see if sufficient disk space is present for the current installation.

Properties

Property	Description
Components	Gets an accessor for components in the current session.
CustomActionData	Gets custom action data for the session that was supplied by the caller.
Database	Gets the Database for the install session.
Features	Gets an accessor for features in the current session.
Item	Gets or sets the string value of a named installer property, as maintained by the Session object in the in-

Property	Description
	memory Property table, or, if it is prefixed with a percent sign (%), the value of a system environment variable for the current process.
Language	Gets the numeric language ID used by the current install session.

Remarks

This object is associated with a standard set of action functions, each performing particular operations on data from one or more tables. Additional custom actions may be added for particular product installations. The basic engine function is a sequencer that fetches sequential records from a designated sequence table, evaluates any specified condition expression, and executes the designated action. Actions not recognized by the engine are deferred to the UI handler object for processing, usually dialog box sequences. Note that only one Session object can be opened by a single process.

`WixToolset.Dtf.WindowsInstaller.dll` version `4.0.2+644ed0118bae1aa885bb6cc906dc6c0b904de4d9`

Dispose(disposing) Method

Closes the session handle. Also closes the active database handle, if it is open. After closing a handle, further method calls may throw an «see T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException» .

Declaration

```
protected void Dispose(
  bool disposing
)
```

Parameters

Parameter	Type	Description
disposing	bool	If true, the method has been called directly or indirectly by a user's code, so managed and unmanaged resources will be disposed. If false, only unmanaged resources will be disposed.

DoAction(action) Method

Executes a built-in action, custom action, or user-interface wizard action.

Declaration

```
public void DoAction(
  string action
)
```

Parameters

Parameter	Type	Description

Parameter	Type	Description
action	string	Name of the action to execute. Case-sensitive.

Remarks

The DoAction method executes the action that corresponds to the name supplied. If the name is not recognized by the installer as a built-in action or as a custom action in the CustomAction table, the name is passed to the user-interface handler object, which can invoke a function or a dialog box. If a null action name is supplied, the installer uses the upper-case value of the ACTION property as the action to perform. If no property value is defined, the default action is performed, defined as "INSTALL". Actions that update the system, such as the InstallFiles and WriteRegistryValues actions, cannot be run by calling MsiDoAction. The exception to this rule is if DoAction is called from a custom action that is scheduled in the InstallExecuteSequence table between the InstallInitialize and InstallFinalize actions. Actions that do not update the system, such as AppSearch or CostInitialize, can be called. Win32 MSI API: MsiDoAction

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:WixToolset.Dtf.WindowsInstaller.InstallCanceledException	the user exited the installation

DoAction(action, actionData) Method

Executes a built-in action, custom action, or user-interface wizard action.

Declaration

```
public void DoAction(
  string action,
  CustomActionData actionData
)
```

Parameters

Parameter	Type	Description
action	string	Name of the action to execute. Case-sensitive.
actionData	CustomActionData	Optional data to be passed to a deferred custom action.

Remarks

The DoAction method executes the action that corresponds to the name supplied. If the name is not recognized by the installer as a built-in action or as a custom action in the CustomAction table, the name is passed to the user-interface handler object, which can invoke a function or a dialog box. If a null action name is supplied, the installer uses the upper-case value of the ACTION property as the action to perform. If no property value is defined, the default action is performed, defined as "INSTALL". Actions that update the system, such as the InstallFiles and WriteRegistryValues actions, cannot be run by calling MsiDoAction. The exception to this rule is if DoAction is called from a custom action that is scheduled in the InstallExecuteSequence table between the InstallInitialize and InstallFinalize actions. Actions that do not update the system, such as AppSearch or CostInitialize, can be called. If the called action is a deferred, rollback, or commit custom action, then the supplied *actionData* will be available via the «see P:WixToolset.Dtf.WindowsInstaller.Session.CustomActionData» property of that custom action's session. Win32 MSI API: MsiDoAction

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:WixToolset.Dtf.WindowsInstaller.InstallCanceledException	the user exited the installation

DoActionSequence(sequenceTable) Method

Executes an action sequence described in the specified table.

Declaration

```
public void DoActionSequence(
  string sequenceTable
)
```

Parameters

Parameter	Type	Description
sequenceTable	string	Name of the table containing the action sequence.

Remarks

This method queries the specified table, ordering the actions by the numbers in the Sequence column. For each row retrieved, an action is executed, provided that any supplied condition expression does not evaluate to FALSE. An action sequence containing any actions that update the system, such as the InstallFiles and WriteRegistryValues actions, cannot be run by calling DoActionSequence. The exception to this rule is if DoActionSequence is called from a custom action that is scheduled in the InstallExecuteSequence table between the InstallInitialize and InstallFinalize actions. Actions that do not update the system, such as AppSearch or CostInitialize, can be called. Win32 MSI API: MsiSequence

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:WixToolset.Dtf.WindowsInstaller.InstallCanceledException	the user exited the installation

EvaluateCondition(condition) Method

Evaluates a logical expression containing symbols and values.

Declaration

```
public bool EvaluateCondition(
  string condition
)
```

Parameters

Parameter	Type	Description
condition	string	conditional expression

Return value

`bool` The result of the condition evaluation

Remarks

Win32 MSI API: MsiEvaluateCondition

Exceptions

Exception	Description

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:System.ArgumentNullException	the condition is null or empty
T:System.InvalidOperationException	the conditional expression is invalid

EvaluateCondition(condition, defaultValue) Method

Evaluates a logical expression containing symbols and values, specifying a default value to be returned in case the condition is empty.

Declaration

```
public bool EvaluateCondition(
  string condition,
  bool defaultValue
)
```

Parameters

Parameter	Type	Description

Parameter	Type	Description
condition	string	conditional expression
defaultValue	bool	value to return if the condition is empty

Return value

`bool` The result of the condition evaluation

Remarks

Win32 MSI API: MsiEvaluateCondition

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:System.InvalidOperationException	the conditional expression is invalid

Format(format) Method

Formats a string containing installer properties.

Declaration

```
public string Format(
  string format
)
```

Parameters

Parameter	Type	Description
format	string	A format string containing property tokens

Return value

`string` A formatted string containing property data

Remarks

Win32 MSI API: MsiFormatRecord

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Record handle is invalid

FormatRecord(record) Method

Returns a formatted string from record data.

Declaration

```
public string FormatRecord(
  Record record
)
```

Parameters

Parameter	Type	Description
record	Record	Record object containing a template and data to be formatted. The template string must be set in field 0 followed by any referenced data parameters.

Return value

`string` A formatted string containing the record data

Remarks

Win32 MSI API: MsiFormatRecord

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Record handle is invalid

FormatRecord(record, format) Method

Returns a formatted string from record data using a specified format.

Declaration

```
public string FormatRecord(
  Record record,
  string format
)
```

Parameters

Parameter	Type	Description
record	Record	Record object containing a template and data to be formatted
format	string	Format string to be used instead of field 0 of the Record

Return value

`string` A formatted string containing the record data

Remarks

Win32 MSI API: MsiFormatRecord

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Record handle is invalid

FromHandle(handle, ownsHandle) Method

Creates a new Session object from an integer session handle.

Declaration

```
public static Session FromHandle(
  IntPtr handle,
  bool ownsHandle
)
```

Parameters

Parameter	Type	Description
handle	IntPtr	Integer session handle
ownsHandle	bool	true to close the handle when this object is disposed or finalized

Remarks

This method is only provided for interop purposes. A Session object should normally be obtained by calling «see M:WixToolset.Dtf.WindowsInstaller.Installer.OpenPackage(WixToolset.Dtf.WindowsInstaller.Database,System.Boolean)» or «see M:WixToolset.Dtf.WindowsInstaller.Installer.OpenProduct(System.String)» .

GetMode(mode) Method

Gets the designated mode flag for the current install session.

Declaration

```
public bool GetMode(
  InstallRunMode mode
)
```

Parameters

Parameter	Type	Description
mode	InstallRunMode	The type of mode to be checked.

Return value

`bool` The value of the designated mode flag.

Remarks

Note that only the following run modes are available to read from a deferred custom action:

- «see F:WixToolset.Dtf.WindowsInstaller.InstallRunMode.Scheduled»
- «see F:WixToolset.Dtf.WindowsInstaller.InstallRunMode.Rollback»
- «see F:WixToolset.Dtf.WindowsInstaller.InstallRunMode.Commit»

Win32 MSI API: MsiGetMode

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:System.ArgumentOutOfRangeException	an invalid mode flag was specified

GetProductProperty() Method

Retrieves product properties (not session properties) from the product database.

Declaration

```
public string GetProductProperty()
```

Return value

`string` Value of the property, or an empty string if the property is not set.

Remarks

Note this is not the correct method for getting ordinary session properties. For that, see the indexer on the Session class. Win32 MSI API: MsiGetProductProperty

GetSourcePath() Method

Gets the full path to the designated folder on the source media or server image.

Declaration

```
public string GetSourcePath()
```

Remarks

Win32 MSI API: MsiGetSourcePath

Exceptions

Exception	Description
T:System.ArgumentException	the folder was not found in the Directory table
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid

GetTargetPath() Method

Gets the full path to the designated folder on the installation target drive.

Declaration

```
public string GetTargetPath()
```

Remarks

Win32 MSI API: MsiGetTargetPath

Exceptions

Exception	Description
T:System.ArgumentException	the folder was not found in the Directory table
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid

GetTotalCost() Method

Gets the total disk space per drive required for the installation.

Declaration

```
public IList<WixToolset.Dtf.WindowsInstaller.InstallCost> GetTotalCost()
```

Return value

`IList<WixToolset.Dtf.WindowsInstaller.InstallCost>` A list of InstallCost structures, specifying the cost for each drive

Remarks

Win32 MSI API: MsiEnumComponentCosts

Log(msg) Method

Writes a message to the log, if logging is enabled.

Declaration

```
public void Log(
  string msg
)
```

Parameters

Parameter	Type	Description
msg	string	The line to be written to the log

Remarks

Win32 MSI API: MsiProcessMessage

Log(format, args) Method

Writes a formatted message to the log, if logging is enabled.

Declaration

```
public void Log(
  string format,
  System.Object[] args
)
```

Parameters

Parameter	Type	Description
format	string	The line to be written to the log, containing 0 or more format specifiers
args	System.Object[]	An array containing 0 or more objects to be formatted

Remarks

Win32 MSI API: MsiProcessMessage

Message(messageType, record) Method

Performs any enabled logging operations and defers execution to the UI handler object associated with the engine.

Declaration

```
public MessageResult Message(
  InstallMessage messageType,
  Record record
)
```

Parameters

Parameter	Type	Description
messageType	InstallMessage	Type of message to be processed
record	Record	Contains message-specific fields

Return value

`MessageResult` A message-dependent return value

Remarks

Logging may be selectively enabled for the various message types. See the «see M:WixToolset.Dtf.WindowsInstaller.Installer.EnableLog(WixToolset.Dtf.WindowsInstaller.InstallLogModes,System.String)» method. If record field 0 contains a formatting string, it is used to format the data in the other fields. Else if the message is an error, warning, or user message, an attempt is made to find a message template in the Error table for the current database using the error number found in field 1 of the record for message types and return values. The *messageType* parameter may also include message-box flags from the following enumerations: System.Windows.Forms.MessageBoxButtons, System.Windows.Forms.MessageBoxDefaultButton, System.Windows.Forms.MessageBoxIcon. These flags can be combined with the InstallMessage with a bitwise OR. Note, this method never returns Cancel or Error values. Instead, appropriate exceptions are thrown in those cases. Win32 MSI API: MsiProcessMessage

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session or Record handle is invalid
T:System.ArgumentOutOfRangeException	an invalid message kind is specified
T:WixToolset.Dtf.WindowsInstaller.InstallCanceledException	the user exited the installation
T:WixToolset.Dtf.WindowsInstaller.InstallerException	the message-handler failed for an unknown reason

SetInstallLevel(installLevel) Method

Sets the install level for the current installation to a specified value and recalculates the Select and Installed states for all features in the Feature table. Also sets the Action state of each component in the Component table based on the new level.

Declaration

```
public void SetInstallLevel(
  int installLevel
)
```

Parameters

Parameter	Type	Description
installLevel	int	New install level

Remarks

The SetInstallLevel method sets the following:

- The installation level for the current installation to a specified value
- The Select and Installed states for all features in the Feature table
- The Action state of each component in the Component table, based on the new level If 0 or a negative number is passed in the iInstallLevel parameter, the current installation level does not change, but all features are still updated based on the current installation level. Win32 MSI API: MsiSetInstallLevel

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid

SetMode(mode, value) Method

Sets the designated mode flag for the current install session.

Declaration

```
public void SetMode(
  InstallRunMode mode,
  bool value
)
```

Parameters

Parameter	Type	Description
mode	InstallRunMode	The type of mode to be set.
value	bool	The desired value of the mode.

Remarks

Win32 MSI API: MsiSetMode

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:System.ArgumentOutOfRangeException	an invalid mode flag was specified
T:System.InvalidOperationException	the mode cannot not be set

SetTargetPath() Method

Sets the full path to the designated folder on the installation target drive.

Declaration

```
public void SetTargetPath()
```

Remarks

Setting the target path of a directory changes the path specification for the directory in the in-memory Directory table. Also, the path specifications of all other path objects in the table that are either subordinate or equivalent to the changed path are updated to reflect the change. The properties for each affected path are also updated. If an error occurs in this function, all updated paths and properties revert to their previous values. Therefore, it is safe to treat errors returned by this function as non-fatal. Do not attempt to configure the target path if the components using those paths are already installed for the current user or for a different user. Check the ProductState property before setting the target path to determine if the product containing this component is installed. Win32 MSI API: MsiSetTargetPath

Exceptions

Exception	Description
T:System.ArgumentException	the folder was not found in the Directory table
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid

VerifyDiskSpace() Method

Checks to see if sufficient disk space is present for the current installation.

Declaration

```
public bool VerifyDiskSpace()
```

Return value

`bool` True if there is sufficient disk space; false otherwise.

Remarks

Win32 MSI API: MsiVerifyDiskSpace

Components Property

Gets an accessor for components in the current session.

Declaration

```
public ComponentInfoCollection Components { get; set; }
```

CustomActionData Property

Gets custom action data for the session that was supplied by the caller.

Declaration

```
public CustomActionData CustomActionData { get; set; }
```

See also

- M:WixToolset.Dtf.WindowsInstaller.Session.DoAction(System.String,WixToolset.Dtf.WindowsInstaller.CustomActionData)

Database Property

Gets the Database for the install session.

Declaration

```
public Database Database { get; set; }
```

Remarks

Normally there is no need to close this Database object. The same object can be used throughout the lifetime of the Session, and it will be closed when the Session is closed. Win32 MSI API: MsiGetActiveDatabase

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid
T:WixToolset.Dtf.WindowsInstaller.InstallerException	the Database cannot be accessed

Features Property

Gets an accessor for features in the current session.

Declaration

```
public FeatureInfoCollection Features { get; set; }
```

Item Property

Gets or sets the string value of a named installer property, as maintained by the Session object in the in-memory Property table, or, if it is prefixed with a percent sign (%), the value of a system environment variable for the current process.

Declaration

```
public string Item { get; set; }
```

Remarks

Win32 MSI APIs: MsiGetProperty , MsiSetProperty

Exceptions

Exception	Description
T:WixToolset.Dtf.WindowsInstaller.InvalidHandleException	the Session handle is invalid

Language Property

Gets the numeric language ID used by the current install session.

Declaration

```
public int Language { get; set; }
```

Remarks

Win32 MSI API: MsiGetLanguage