SQL Server 2005 - SQL Server Integration Services - Part 9 - WMI Data Reader

Following an extended review of various types of Foreach Loop enumerators in the SQL Server 2005 Integration Services, we are turning our attention to other types of package components, which either did not exist in the Data Transformation Services (SSIS predecessor present in earlier versions of the SQL Server) or which functionality has been significantly enhanced. In this article, we will discuss a task called WMI Data Reader, leveraging functionality built into Windows Instrumentation Management (WMI) technology. As you might already have noticed, SSIS includes also a similar component named WMI Event Watcher Task. While both have some common characteristics (most notably, dependence on WMI and WQL - which we will describe in more details next), we will postpone presentation of the latter one until we cover general concept of events in SSIS packages.

Windows Management Instrumentation is a software component incorporated in more recent releases of the Windows operating system (prior to Windows 2000, it was available as a freely downloadable add-in). Its main goal is to provide the ability to extract and control characteristics of virtually every aspect of the computing environment. WMI has dominated the area of Windows management in recent years, extending its reach into SQL Server administration (for an introduction to this topic, refer to the series of articles "Automating SQL Server Management with WMI" published on the Database Journal Web site). You can take advantage of the powerful capabilities offered by WMI in several ways. The most rewarding (but also most demanding in terms of skills and time required) utilizes a programming and scripting interface, available from both .NET and legacy code (which you can implement via Script Tasks in SQL Server Integration Services packages). However, Microsoft also ensured that a plethora of WMI features is readily accessible even to those programmatically-challenged and created such user-friendly methods of accessing WMI functionality as WMIC (WMI command line utility, introduced in Windows XP and present in Windows 2003 Server) or WMI Tools, including intuitive in use WMI CIM Studio, WMI Object Browser, WMI Event Registration Tool, and WMI Event Viewer (freely downloadable from the Microsoft Download Center). Inclusion of WMI Data Reader and WMI Event Watcher tasks in the SQL Server 2005 Integration Services is another example of the same trend.

Before we start looking into WMI Data Reader Task and its configuration settings, we need to first review the basic principles of WMI as well as the elementary rules of a technique that we will use as the primary means to interact with it, called WMI Query Language (WQL). On the conceptual level, each managed component that falls within the scope of WMI is represented by a class. For example, a class called Win32_OperatingSystem deals with properties and functions of the operating system on a managed computer. Similarly, there exist such classes as Win32_Service, Win32_Process, Win32_Processor, Win32_NetworkAdapter, Win32_LogicalDisk, Win32_Directory, or Win32_NTLogEvent (representing services, processes, processors, network adapters, logical disks, file system directories, and Windows events). The sheer number of classes is overwhelming (and still growing, since WMI is extensible, allowing Microsoft and third party vendors to develop new ones). In order to improve their manageability, classes are organized into a hierarchy (which accommodates such programming-friendly features as inheritance) and separated, based on similarities and vendor responsible for their creation, into namespaces (which simplify searches, prevent accidental class name clashes, and provide security boundaries). Classes we will work with throughout our articles reside in the root\cimv2 namespace (where the majority of standard physical and logical computer components have their WMI presence) and

root\MicrosoftSQLServer namespace (where all SQL Server specific classes are located). If you want to gain a better understanding of the way namespaces and classes are arranged, browse through their hierarchy with WMI CIM Studio (via Internet Explorer interface), which is part of the earlier mentioned WMI Tools.

A class does not represent any particular object, but rather a general concept of a managed component. For example, the class Win32_LogicalDisk does not refer to a specific drive, but instead encompasses features inherent to every type of logical disk, namely properties (which values can be read and, sometimes, changed - for example file system, media type, volume name, etc.) and methods (which describe types of actions that can be performed on a disk - e.g. running Chkdsk or scheduling AutoChk). Classes that are of interest to us typically have one or more instances (which are also referred to as objects) on a managed system (e.g. logical disk C: is an object of Win32_LogicalDisk class). Most properties of an object have their values set and the execution of a method on an object produces a specific action.

WQL is a subset of ANSI SQL (standard established by American National Standards Institute). While its basic syntax resembles equivalent T-SQL statements, there are several differences, resulting from characteristics inherent to WMI (for a list of WQL keywords and operators, refer to the MSDN Web site). Similar to T-SQL, its main purpose is data retrieval, although this is accomplished through interaction with WMI and the data itself represents the state of a managed system. Most commonly, information about an instance of a class is extracted with the familiar SELECT statement (while there are other types of queries, such as ASSOCIATORS OF, REFERENCES OF, or statements which deal with class definitions, they are outside of the scope of this article), which takes the following form:

```
SELECT PropertyA, PropertyB, PropertyC FROM ClassX WHERE PropertyD =
"ValueY"
```

As you would expect, in the statement above, we want to obtain values of PropertyA, PropertyB, and PropertyC of an instance (or instances, if there are multiple matches that satisfy the WHERE clause) of ClassX, identified by the ValueY of the PropertyD. For example, in order to determine values of FileSystem, FreeSpace, and Size properties of the Win32_LogicalDisk class instance named "C:", we would execute:

```
SELECT FileSystem, FreeSpace, Size FROM Win32_LogicalDisk WHERE Name="C:"
```

Similarly, we can query properties of services (Caption or DisplayName, StartMode, and State of Win32_Service class) or processes (ExecutablePath and Name of the Win32_Process class) in order to determine whether a specific service is running or a specific process has been launched. WMI also gives you access to Windows Event logs through Win32_NTLogEvent class, which instances represent individual entries from System, Security, and Application logs (to distinguish between them, check the value of the Logfile property for each). Here is a sample query that returns the relevant information:

```
SELECT Logfile, RecordNumber, ComputerName, EventCode, EventIdentifier,
Message, SourceName, TimeGenerated, Type, User
FROM Win32_NTLogEvent
```

By checking values assigned to the properties of instances of the Win32_QuickFixEngineering class (including ServicePackInEffect, HotFixID, and InstaledOn), it is possible to determine the level of hotfixes on a target computer.

Instances of another class, Win32_Product, represent every Windows Installer-based application present on a target system. As long as your software deployment methodology relies on Windows Installer (which applies to the overwhelming majority of newer software), you can obtain accurate information about your application estate, simplifying compliance with licensing requirements and ensuring that no rogue, unauthorized programs exist in your environment. The query providing this type of information could resemble the following:

```
SELECT Name, Version, Description, InstallDate, Vendor, InstallLocation
FROM Win32_Product
```

Equipped with this knowledge, let's look at the configuration parameters of the WMI Data Reader Task. Like other Control Flow elements, this task appears in the Toolbox of the package designer area. As mentioned before, its main purpose is to execute a WMI query (using WQL syntax) and return results according to the following options available in the task's Editor interface (in the WMI Options section):

- WmiConnection - determines the target system and WMI namespace (which, by default is, root\cimv2), as well as the authentication method applied when connecting to it. If you decide to apply Windows Authentication, keep in mind that the connection will be established in the security context in which the SSIS package is executing (which might vary, depending on whether the package is launched interactively or scheduled). Ensure that the account has sufficient permissions as far as DCOM and WMI-namespace are concerned (for a comprehensive listing of issues and respective resolutions that involve remote access to WMI on Windows XP SP2 systems, refer to the Microsoft Knowledge Base article KB 875605).
- WqlQuerySourceType - can be set to Direct input, File connection, or Variable. Obviously, the last two options require the creation of the appropriate file connection and variable, respectively.
- WqlQuerySource - the type of value expected for this entry depends on the value of the WqlQuerySourceType. If you intend to test the WQL queries listed in this article, consider selecting Direct input, and copying individual queries into the WQL Query window.
- OutputType - list of choices include Data table, Property name and value, and Property value. The choice depends primarily on the purpose of extracting WMI data and its final destination.
- OverwriteDestination - here, you can specify how you want to handle a situation where the target store already contains some data (overwrite destination, keep original, or append to existing content).
- DestinationType - this can be either a file connection or a variable.
- Destination - value of this entry depends on the previous entry and requires either a variable or a file connection to be defined.

In the next article of this series, we will review the role of events in SQL Server Integration Services, which will serve as the basis for the discussion about WMI Event Watcher Task.