[](http://www.sqlbi.com)

We [solve complex problems](http://new.sqlbi.com/Default.aspx#.html) of [data modeling](http://new.sqlbi.com/Default.aspx#.html) and [develop tools](http://new.sqlbi.com/Default.aspx#.html) and solutions to let business perform best through data analysis

**Description of a tool: SQLBI DTEXEC**

*SqlBiDtExec is a tool that makes it easy to launch SSIS packages on a remote server, using WCF as the transport media to both request package execution and fetch all the logs and events from the package in order to use them at the client side.*

The DTEXEC utility provided by Microsoft to execute SSIS packages is only able to execute them locally and provides no support to execute a package on a remote server. For this reason, we wrote a WCF service that is able to launch SSIS packages using the DTS runtime library and send back results to the caller through the WCF callback system. The WCF service can be hosted in Internet Information Services, in a Windows service or in a console application with a very small effort. Moreover, using the standard WCF communication method, it can be very easily configured to work in any network environment.

The solution is composed of many projects, all in the SqlBi.DtExec namespace :

* Classes: this class library contains the main classes of the projects, which are used both from the client and the server side.
* ClientAgent: this class library contains a single class, ClientAgent, which is an easy interface that can be used to communicate with the service, without the need to open channels and handle all the complex WCF methods.
* Contracts: this class library contains the WCF contract interface between the client and the server.
* HostingConsole: this sample application hosts the WCF service in a console program.
* HostingService: this sample Windows service application hosts the WCF service in a windows service and can be used on any server to make it able to answer DtExec requests.
* Services: this is the “real” implementation of the WCF service, it implements the contract interface and answer to DtExec requests.
* TestApplication: this sample application uses the ClientAgent to communicate with a server and remotely execute packages. It can be configured through an XML file and writes the log and the event flows on files and/or on console.

The configuration of the WCF endpoint can be done in both the client and server applications using the app.config file. Please note that the service configuration file has a different file name (SqlBI.DtExec.HostingService.config) since, being a service, it needs a different configuration file name.

We are not going to describe the app.config entries, since they are standard WCF configurations and their complete description is far outside the scope of this document. Moreover, we are not going in a deep explanation of the solution itself, in this document we are only providing a simple description of the architecture, urging the interested reader to study the source code.

# The configuration

Each package to run is contained in a PackageInformation class, which holds all the information needed to execute a package. The parameters are pretty easy:

* Name: a name for the package.
* EndPointToUse: contains the name of the WCF endpoint to use in order to execute the package. By using different endpoints a single client can call different services on different servers.
* Path: contains the path (relative to the executing server) of the package to execute.
* Password: if the package is password protected, then provide here the password to open it.
* LogToFileName: a path, relative to the client, where to write the log entries.
* LogToConsole: a boolean value, if true the log will be sent on the console
* EventsToFileName: a path, relative to the client, where to write the eventsentries.
* EventsToConsole: a boolean value, if true the events will be sent on the console
* Configurations: here you can provide package configurations. The values provided will be set as values of the configurations.
* Logging: here you can provide some logging options, i.e. which information to send back to the client (like user name, computer name and so on)
* Events: here you can provide some events options, i.e. the kind of events you want to receive from the package execution.

The sample configuration file in the test application can be used as a pattern. A separate section (DefaultEvents, DefaultLogging) provide a convenient way to define default options for events and logging, which will be used if no specific option is defined for a package. Please note that some configuration are used at client side, while others are used by the server to determine the package behavior. Take a look at the code, to get a better understanding of them.

# The WCF Communication System

The solution uses two contracts:

* IServiceSqlBIDtExec: receives a PackageInformation class and executes it.
* ILogCallback: used to send back information about the log and events. It sends back a LogItem class, which contains the log type, the timestamp of the generation and a string, containing the log message

Both interfaces are used by the Service that implements the DtExec. The client does not need to interface directly with these interfaces, since the ClientAgent manages all the operations, providing a simpler interface to the caller.

ClientAgent provides a very simple interface: ExecutePackage that receives a PackageInformation class and a callback method that will be invoked every time a log is received from the server. Using ClientAgent it is very simple to configure a PackageInformation class, write a callback handler (which can do whatever it wants to handle logs) and then execute remotely the package.

# Conclusion

This solution is a perfect example of the power of WCF. All the complex mechanism of communication between the client and the server is handled by WCF, and the configuration of both sides of the communication layer are freely configurable via the app.config program, which makes it very easy to adapt the system to any network environment.

The software as is might not be perfect for your specific needs, nevertheless it is very easy to update to accommodate any need. If, for any reason, you implement some interesting feature, we’d like to know it and receive the updated sources, so that we can integrate new features in the public version of the source code.