Database Connectivity Guide for TestExec SL

Application Note

Overview

The requirement for database storage is gaining prevalence in today's industries as manufacturing becomes more complex. Essential information such as results of unit under test (UUT) should be stored for further data analysis or as part of operations records.
This application note outlines the importance of proper data logging in a database and discusses best practices to import extensible markup language (XML) files from TestExec into a database. This document assumes you have working knowledge of database concepts, Structured Query Language (SQL) and database management system (DBMS).

Introduction

In industries where automated test systems, functional tests and design verifications play a key role, the importance of data logging is irrefutable. The data generated from a test execution can be formatted for storage in a spreadsheet or database via the TestExec's data configuration editor (DCE). While there are alternative ways of data storage, it all boils down to the users' needs and preferences. In high volume manufacturing where new data is generated every few seconds, it is recommended that a database is set up for fast and efficient data retrieval. The advantages of database storage include:

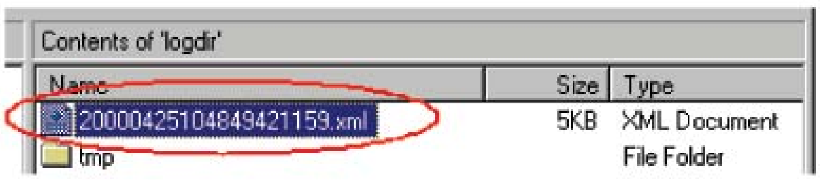
- Fast and efficient data retrieval – A database helps you to organize data in a logical manner and the database management systems are fine-tuned to rapidly retrieve the data just the way you want it. In addition, databases help you to break your data into specific parts and you only have to specify the right SQL code to extract your data as compared to scrutinizing every folder to find a particular XML file that contained the record you wanted.
- Centralized storage – For an organization with multiple manufacturing sites around the world, engineers are able to access data linked up via a networked database. This is a significant benefit for teams that are working on the same project and can simultaneously access the database at the same time.

Agilent Technologies

- Security – To access a database, each user has to log on as a specific user and each user has various rights and limits. Most database management systems allow you to configure a user with its corresponding level of security, such as full access to the database administrator to change the database's structure or to delete users as well as only read access for operators.
- Maintenance – With enormous amounts of data that need to be searched, sorted and regularly updated, databases combined with SQL allow you to get the data in the order you want it.

1.0 Overview on Data Logging in Test Exec SL

Each time a testplan is run, the corresponding logged data is stored in your selected data logging directory. By default, the location is \My TxSL Files\Log Files.
Each file has a unique name and an extension of ".xml" as the default data is written in industry-standard XML format. An example of a data logging file in the log directory is shown below.

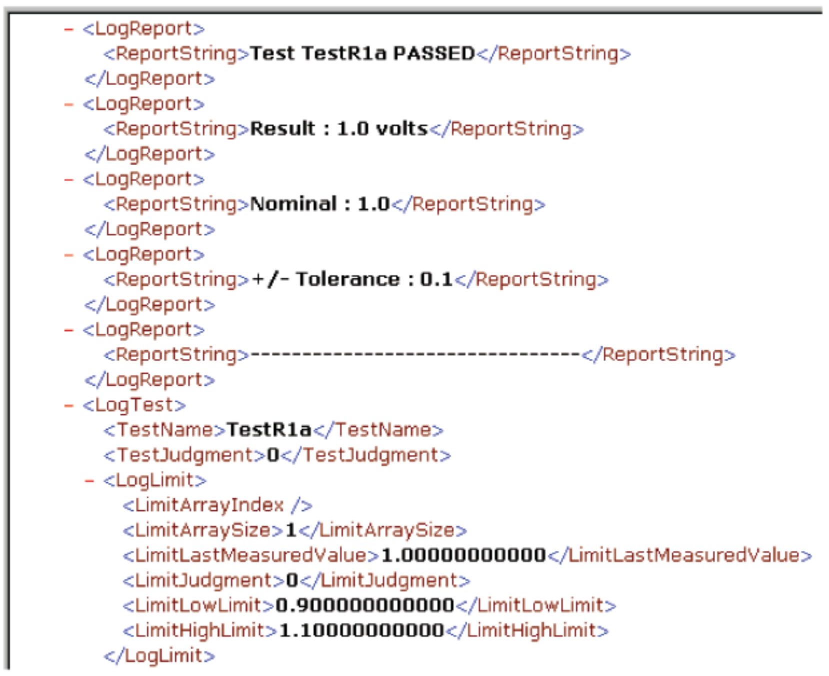


Contents of 'logdir'

Name	Size	Type
20000425104849421159.xml	5KB	XML Document
tmp		File Folder

Figure 1: Example of data logging file

The XML file contains data resulting from the execution of testplans and is customizable by user. An excerpt from the data logging file for a sample test-plan is shown in Figure 2:



```
- <LogReport>
    <ReportString>Test TestR1a PASSED</ReportString>
  </LogReport>
- <LogReport>
    <ReportString>Result : 1.0 volts</ReportString>
  </LogReport>
- <LogReport>
    <ReportString>Nominal : 1.0</ReportString>
  </LogReport>
- <LogReport>
    <ReportString>+/- Tolerance : 0.1</ReportString>
  </LogReport>
- <LogReport>
    <ReportString>--------------------------------</ReportString>
  </LogReport>
- <LogTest>
    <TestName>TestR1a</TestName>
    <TestJudgment>0</TestJudgment>
  - <LogLimit>
      <LimitArrayIndex />
      <LimitArraySize>1</LimitArraySize>
      <LimitLastMeasuredValue>1.00000000000</LimitLastMeasuredValue>
      <LimitJudgment>0</LimitJudgment>
      <LimitLowLimit>0.900000000000</LimitLowLimit>
      <LimitHighLimit>1.10000000000</LimitHighLimit>
    </LogLimit>
```

Figure 2: Contents in data logging file

The data resulted from the execution of testplan as illustrated in Figure 2 (in Bold) is enclosed in a nested structure of identifiers that provide structure for the data. This makes it possible for other programs to parse the data into its individual components later. You could write a custom program to parse it for use with your favorite database or statistical quality control tool.

2.0 Setting Configuration in the Data Configuration Editor (DCE)

You can specify the type of records or fields to appear in the data logging file via Data Logging Configuration Editor. In addition, you can define the attributes of the records and fields as well as specify the behavior and format for the data logging files.

Figure 3 is a snapshot of the Data Logging Configuration Editor. On the left pane is the ‘Record Hierarchy’, which is a hierarchy of log records and the corresponding fields. The right pane consists of ‘Field Definitions from..’ which lists all possible fields that you can associate with the records in the ‘Record Hierarchy’.

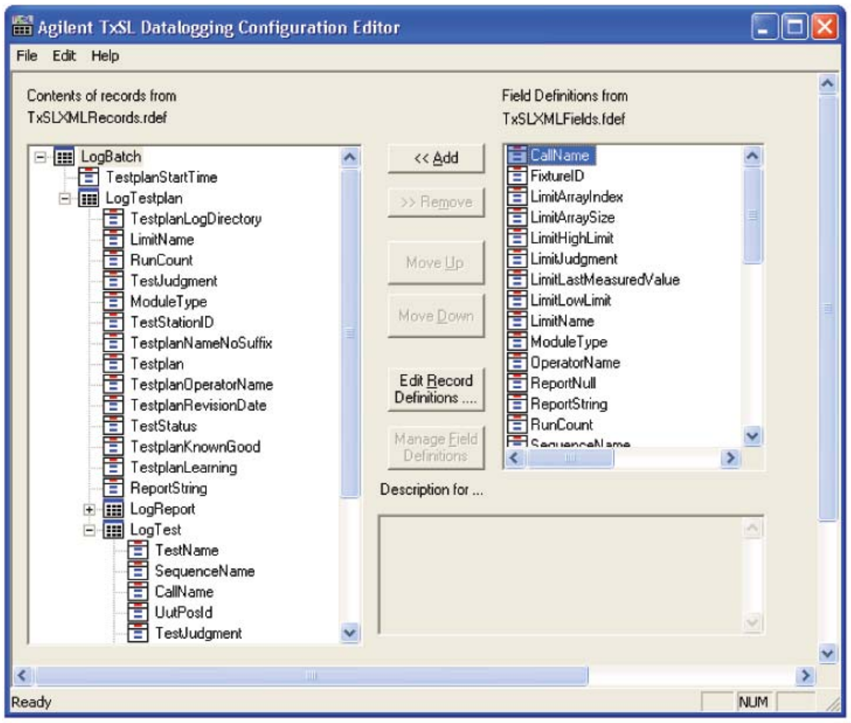


Figure 3: Data Logging Configuration Editor

3.0 Database Concepts

A database is a structured collection of records or data that is stored in a computer system. Every table contains records known as rows and their corresponding fields, also known as columns. Each column has a data type and this is configured using the DBMS.

Table 1 shows an example of how a database can store results for testing purposes.

UUT Serial Number	Test Name	Test Status
301C098	TEST1	PASS
301C060	TEST2	FAIL
301C567	TEST1	PASS

Table 1: Example of Database Table

4.0 Importing XML results from TestExec into Microsoft SQL Server

There are a few methods to import XML file into Microsoft ® SQL Server, such as by using the OPENXML, T-SQL, SSIS or XML Bulk Load component. However, OPENXML and T-SQL have a common drawback – they are not suitable for loading large amounts of data, which results in slow processing and is resource intensive.

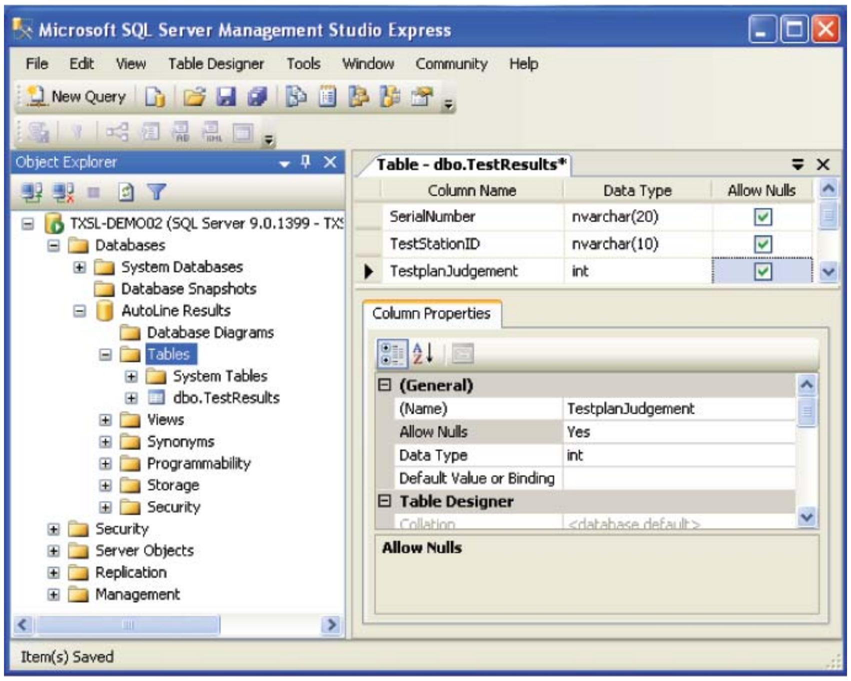
SQLXML provides the facility intended specifically to address this problem. Called the XML Bulk Load component, it is a COM component that you can call from OLE Automation-capable languages and tools comprising Visual Basic, Delphi and Transact-SQL. The OLE Automation-capable languages include Windows ® scripting languages such as VBScript and JScript.

This document focuses on using the XML Bulk Load component to insert XML data into SQL and assumes that you have completed a full installation of Microsoft SQL Server 2005 that includes SQLXML 4.0.

Step 1: Creating table in Microsoft SQL Server 2005

- Connect to the SQL Server and create a database with the title 'AutoLine Results'.
- In the 'AutoLine Results' database, create a table with the title 'TestResults' and its corresponding columns. Alternatively, you can run the following SQL statement in Query Analyzer:

```
USE AutoLine Results
CREATE TABLE TestResults (
  SerialNumber NVARCHAR(20),
  TestStationID NVARCHAR(10),
  TestplanJudgement INT(10),
```



Column Name	Data Type	Allow Nulls
SerialNumber	nvarchar(20)	✓
TestStationID	nvarchar(10)	✓
TestplanJudgement	int	✓

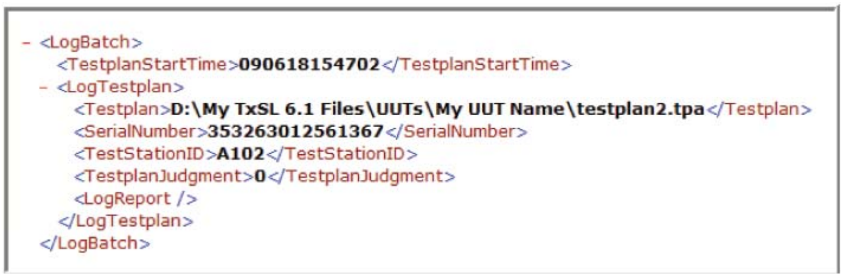
(General)	
(Name)	TestplanJudgement
Allow Nulls	Yes
Data Type	int
Default Value or Binding	
Table Designer	
Collation	<database default>

Allow Nulls

Figure 4: Creating Table

Step 2: Configure the format for the XML data logging file

As mentioned in Section 2, you can configure the format of the XML output using the DCE. For the example here, three parameters are identified – SerialNumber, TestStationID and Testplan Judgement. By default, the filename should be in timestamp format but it has been changed to string format in this document for clarity purposes. Figure 5 shows a sample of XML file (C:\ TestResults.xml).



```
- <LogBatch>
    <TestplanStartTime>090618154702</TestplanStartTime>
  - <LogTestplan>
      <Testplan>D:\My TxSL 6.1 Files\UUTs\My UUT Name\testplan2.tpa</Testplan>
      <SerialNumber>353263012561367</SerialNumber>
      <TestStationID>A102</TestStationID>
      <TestplanJudgment>0</TestplanJudgment>
      <LogReport />
    </LogTestplan>
  </LogBatch>
```

Figure 5: XML file from TestExec

Step 3: Create the mapping schema file

The first step in using the XML Bulk Load component is to define a mapping schema that maps the XML data from TestExec into the tables and columns in your database. When the component loads the XML data, it will read the data as a stream and use the mapping schema to decide where exactly the data should go in the database.

This is a sample mapping schema that maps the format of TestResults.xml to the format of the TestResults table in the ‘AutoLine Results’ database. Paste this schema into ‘Notepad’ and save the file as C:/Testresultsmapping.xml.

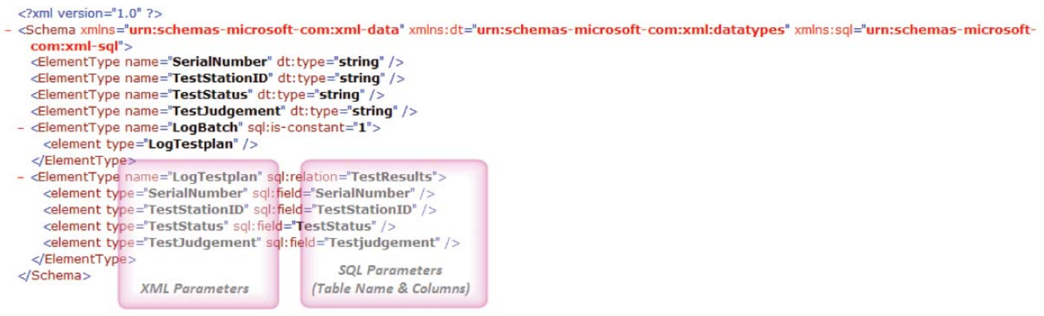
•

```
<?xml version=”1.0” ?>
<Schema xmlns=”urn:schemas-microsoft-com:xml-data”
        xmlns:dt=”urn:schemas-microsoft-com:xml:datatypes”
        xmlns:sql=”urn:schemas-microsoft-com:xml-sql” >

  <ElementType name=”SerialNumber” dt:type=”string” />
  <ElementType name=”TestStationID” dt:type=”string” />
  <ElementType name=”TestplanJudgement” dt:type=”int” />

  <ElementType name=”LogBatch” sql:is-constant=”1”>
    <element type=”LogTestplan” />
  </ElementType>

  <ElementType name=”LogTestplan”  sql:relation=”TestResults”>
    <element type=”SerialNumber”  sql:field=”SerialNumber” />
    <element type=”TestStationID” sql:field=”TestStationID” />
         <element type=”TestplanJudgement”
sql:field=”Testplanjudgement” />
  </ElementType>
```



```
<?xml version="1.0" ?>
- <Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:xml:datatypes" xmlns:sql="urn:schemas-microsoft-
  com:xml-sql">
  <ElementType name="SerialNumber" dt:type="string" />
  <ElementType name="TestStationID" dt:type="string" />
  <ElementType name="TestStatus" dt:type="string" />
  <ElementType name="TestJudgement" dt:type="string" />
- <ElementType name="LogBatch" sql:is-constant="1">
    <element type="LogTestplan" />
  </ElementType>
- <ElementType name="LogTestplan" sql:relation="TestResults">
    <element type="SerialNumber" sql:field="SerialNumber" />
    <element type="TestStationID" sql:field="TestStationID" />
    <element type="TestStatus" sql:field="TestStatus" />
    <element type="TestJudgement" sql:field="Testjudgement" />
  </ElementType>
</Schema>
```

Figure 6: Mapping Schema File

Step 4: Create a VBScript program to execute the XML Bulk Load component

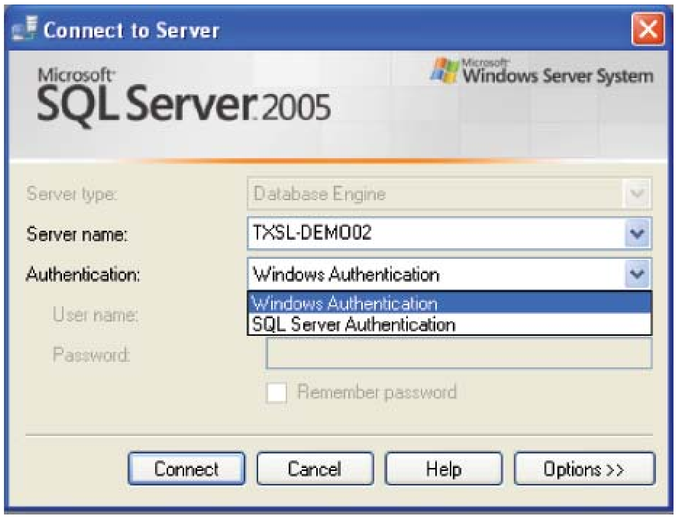
This is the executable script that uses the XML Bulk Load component to insert the three records from the XML file into the database table created by using the mapping schema. Paste this VBScript code into Notepad, and then save the file as C:\Inserttestresults.vbs.

```
Set objBL = CreateObject("SQLXMLBulkLoad.SQLXMLBulkLoad")
objBL.ConnectionString = "provider=SQLOLEDB.1;data source=MySQ
LServer;database=MyDatabase;uid=MyAccount;pwd=MyPassword"
objBL.ErrorLogFile = "c:\error.log"
objBL.Execute "c:\TestResultsMapping.xml", "c:\TestResults.xml"
Set objBL = Nothing
```

Note: If you have connected to the SQL database using the **SQL Server Authentication**, you must configure the 'ConnectionString' credentials (as highlighted) for the script to work with your SQL Server installation. If not configured, the following error message will occur after you execute the script: 'Error connecting to the data source'

Alternatively, if you have connected to the SQL database with your **Windows Authentication** (as shown in Figure 7) you can use this script:

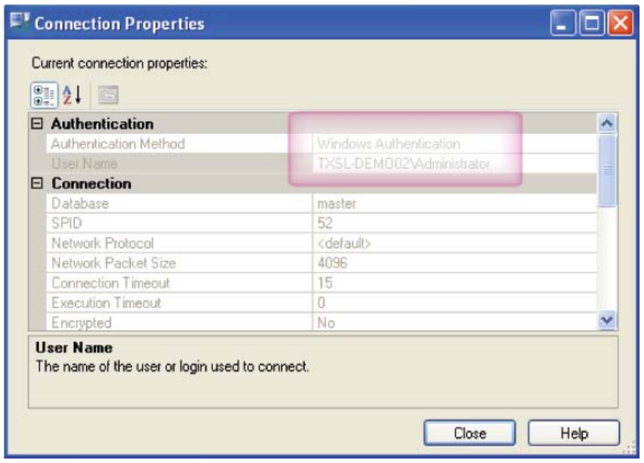
```
Set objBL = CreateObject("SQLXMLBulkLoad.SQLXMLBulkload.4.0")
objBL.ConnectionString = "provider=SQLOLEDB.1;data
source=TXSL-DEMO02;database=AutoLine Results;Integrated
Security=SSPI"
objBL.ErrorLogFile = "c:\error.log"
objBL.Execute "C:\TestResultsMapping.xml", "c:\TestResult.xml"
Set objBL = Nothing
```



Server type: Database Engine
Server name: TXSL-DEMO02
Authentication: Windows Authentication
Windows Authentication
SQL Server Authentication
User name:
Password:
Remember password

Figure 7: Connection Method

Note: Enter your Windows credentials and the database details (as highlighted). You can refer to the Connection Properties for the credentials as shown in Figure 9.



Current connection properties:

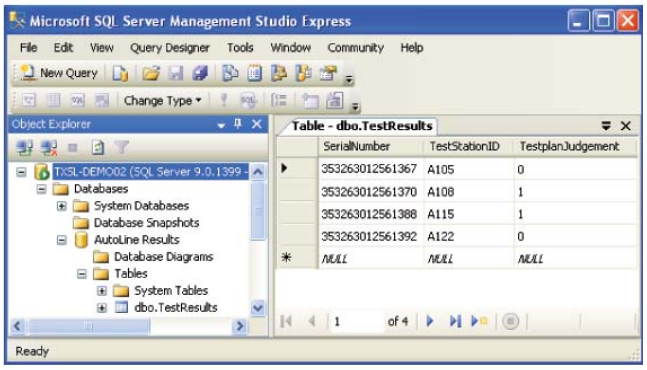
Authentication	
Authentication Method	Windows Authentication
User Name	TXSL-DEMO02\Administrator
Connection	
Database	master
SPID	52
Network Protocol	<default>
Network Packet Size	4096
Connection Timeout	15
Execution Timeout	0
Encrypted	No

User Name
The name of the user or login used to connect.

Figure 8: Connection Properties

Step 5: Run the VB Script program

Double click to run the VB Script program (C:\Inserttestresults.vbs) in order to insert records from XML file into the TestResults table. You will see the data in the database as show in Figure 9.



SerialNumber	TestStationID	TestplanJudgement
353263012561367	A105	0
353263012561370	A108	1
353263012561388	A115	1
353263012561392	A122	0
NULL	NULL	NULL

Figure 9: Results in Database

Note: To automate the VB script for periodic stream of data insertion, you can go to Control Panel and click on Scheduled Tasks. You can then set the scheduler to execute the VB Script on a periodic basis. However, the VB Script must be modified to always detect for new XML files each time it is executed.

Conclusion

In addition to VB Script, you can also consider using VB or .NET for connecting to Microsoft SQL. TestExec offers you the flexibility of customizing your selection of database connectivity. Besides Microsoft SQL, you can utilize other databases such as Oracle® or IBM DB2.

www.agilent.com

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Americas	
Canada	(877) 894-4414
Latin America	305 269 7500
United States	(800) 829-4444

Asia Pacific	
Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe & Middle East	
Austria	01 36027 71571
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700*
	*0.125 €/minute
Germany	07031 464 6333
Ireland	1890 924 204
Israel	972-3-9288-504/544
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland	0800 80 53 53
United Kingdom	44 (0) 118 9276201

Other European Countries:
www.agilent.com/find/contactus

Revised: July 2, 2009

Product specifications and descriptions in this document subject to change without notice.

© Agilent Technologies, Inc. 2009
Printed in USA, July 16, 2009
5990-4367EN

Agilent Technologies