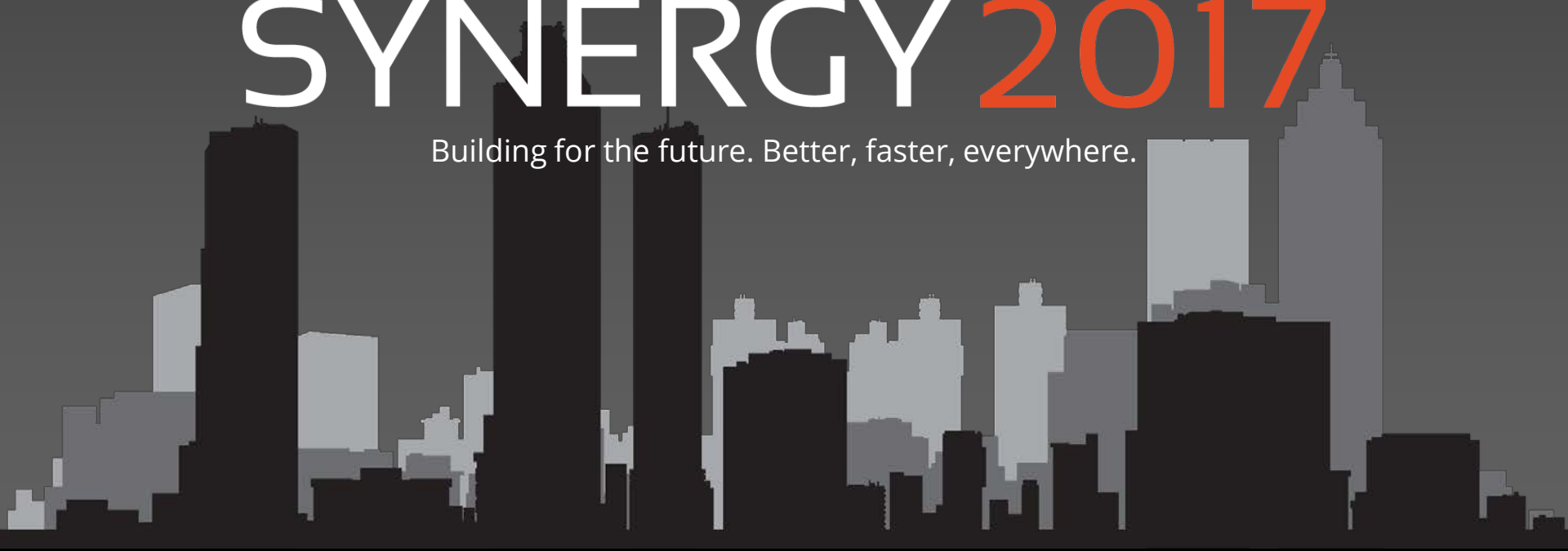
6 – 7 . APRIL ATLANTA, GA, USA

SYNERGY 2017

Building for the future. Better, faster, everywhere.



Introducing SQL Managed Connections

John Tuohy

SQL Connection Basics

- Establishing a connection is how you access SQL data
- This is done with a connection string
- A connection string defines
  - The database server
  - A default database
  - Login credentials
  - And more
- You connect to a database by logging in
- Once you are logged you can access your tables

DataFlex SQL Connection Basics

- In DataFlex the connection string was originally defined in an INT file.

```
DRIVER_NAME MSSQLDRV
SERVER_NAME SERVER=.\SQLEXPRESS;Trusted_Connection=yes;DATABASE=Chinook
DATABASE_NAME Album
SCHEMA_NAME dbo
```

- An INT file is defined for each table in your application
  - Each table can use different connection strings but this is rare
  - The server string information is duplicated across many INT files

- These connections strings are needed to open your application and also needed when using the tools (Studio, Database Builder and Explorer, etc.) and when using Embedded SQL

- If you need to change a connection string, you need to alter that string in all of your INT files. No fun.

Connection Ids

- Connection IDs were introduced to our drivers long ago
- Connection IDs are simply an abstract way to refer to these connections by Id while defining the actual connection string elsewhere
- The Connection ID is used in table INT files...

```
DRIVER_NAME MSSQLDRV
SERVER_NAME DFCONNID=Chinook
DATABASE_NAME Album
SCHEMA_NAME dbo
```

- The connection Id mapping to actual database servers was defined “somewhere else”
- Connection Ids were hard to use
  - They were difficult to configure and maintain
  - They were not directly supported by the DataFlex Framework
  - They were not directly supported by the DataFlex tools (Studio, Database Explorer, etc.)

Introducing Managed Connections

- Connection information is stored at the workspace level in a configuration file
  - The solution is workspace centric
  - The file is a simple .ini file
  - Normally stored in data\dfconnid.ini
- Here the full connection string and credential information is defined in DFConnID.ini

```
[connection1]
id=Chinook
driver=MSSQLDRV
connection=SERVER=.\SQLEXPRESS;DATABASE=Chinook
trusted_connection=yes
```

- The Connection ID is then used in table INT files…

```
DRIVER_NAME MSSQLDRV
SERVER_NAME DFCONNID=Chinook
DATABASE_NAME Album
SCHEMA_NAME dbo
```

- Managed Connections build on existing Connection ID concepts; think of this like “Connection IDs 2.0”

cConnection class

- Managed connections are implemented via the cConnection class.
- The cConnection class will handle all connections for DataFlex CLI drivers (6.2 and higher)
- cConnection is a class that creates a single, global object that allows you to
  - Create and maintain Connection IDs
  - Use Connection IDs in your table INT files
  - Define connections IDs in a workspace connections .ini file
  - Login to database servers via Connection IDs
  - Make ESQL connections to servers via Connection IDs

Managed Connections – The Tools

- We also provide high level tools to configure manage this entire process
  - The Studio, Database Explorer, the other tools and the wizards allow you to create and manage the connections
  - It's easy to create tables using managed connections
- Don't worry about understanding how all the configurations are formatted
  - The tools handle this for you

Managed Connections in your code

- The managed connection is all DataFlex code based
  - Access to this configuration file is handled through a single cConnection object
  - Your applications and our tools, use the same cConnection API
  - It requires very little code to implement in your application

- Code required to support managed connections in application

```
Object oApplication is a cApplication

    Object oConnection is a cConnection
        Use LoginEncryption.pkg
        Use DatabaseLoginDialog.dg
    End_Object

End_Object
```

Supporting additional connections

- You can define more than one connection in the connections .ini file
  - Multiple connections
  - Alternate connections
- Multiple connections are defined when your application needs to open tables from multiple servers
  - Each server will have it’s own ID
- Alternate connections are defined when you wish to run an application using an alternate server
  - The IDs will be the same but only one will be enabled

A connection with multiple connections

```
[connection1]
Id=ID1
driver=MSSQLDRV
connection=SERVER=.\SQLEXPRESS;DATABASE=Order
trusted_connection=yes
[connection2]
Id=RS1
driver=MSSQLDRV
connection=SERVER=MyRemoteServer;DATABASE=RemoteData
UID=AppUser
PWD=893753hskfgd
```

A connection with alternate connections

```
[connection1]
Id=ID1
driver=MSSQLDRV
connection=SERVER=.\SQLEXPRESS;DATABASE=Order
trusted_connection=yes
disabled=yes
[connection2]
Id=ID1
driver=MSSQLDRV
connection=SERVER=.\SQLEXPRESS;DATABASE=Order_Demo
trusted_connection=yes
```

Managed Connections, Encryption and Database Logins

- An application needs to login into a database server. Usually this
  - occurs when the application is started
  - is required – if login fails, the application should not be run
  - is silent - it does not require user interaction
  - uses credential information stored with the application's configuration data (dfconnid.ini file)
  - The stored credential information must be secure
  - Note: this is not a user login - this occurs *before* a user login
- Managed Connections handles all of this

Storing Login Credentials

- Storing encrypted passwords creates some challenges
  - This must be supported both for your applications and in our tools
    - The Application encryption method should be fully customizable and only known by the developer
    - The Tool encryption method is controlled and only known by us
- We solve this by storing two password encryptions
  - PWD – this stores the application password encrypted using a method known only to the application developer
  - DFPWD – this stores our Studio (and tools) password encrypted using a method know only to us

The Connections .ini file

- A connection with user id / password information

```
[connection1]
id=Chinook
driver=MSSQLDRV
connection=SERVER=.\SQLEXPRESS;DATABASE=Chinook
UID=AppUser
PWD=8973753hskfjd
DFPWD=sdfj876jdk
```

The Database Login Tool

- A tool is required to configure the credential information.
- That tool is a database login dialog that
  - is only invoked when needed
  - accepts input to perform the login
  - stores the successful credentials
  - uses the applications encryptions rules to store passwords
  - We provide you that tool
  - It uses a workspace unique random key to seed the encryption and it can be further customized by the developer
  - can be embedded in your windows application or used standalone
- Our applications (Studio, etc.) uses a similar tool and technique

Encryption and login object packages

- Your application contains code in two object packages that manage encryption and logging in. The standard packages are

```
Object oApplication is a cApplication

    Object oConnection is a cConnection
        Use LoginEncryption.pkg
        Use DatabaseLoginDialog.dg
    End_Object

End_Object
```

- You can replace these with your own custom packages using ours as your template.

Dynamic connections

- cConnection makes it possible to change database servers and databases dynamically
- Applications can select their connection upon startup
- Applications can change their database within a server, while running
- Applications can redirect their server/database connections at runtime
- Paves the way for multi-tenant applications
- This will be particularly useful for web applications

Compatibility

- Managed connections are supported with our MSSQL, DB2 and ODBC connectivity kits (starting with revision 6.2)
- The changes have been implemented while maintaining 100% backwards compatibility with...
  - Existing SQL-based applications
  - Embedded database applications
  - Pervasive database applications
- No changes are required to keep doing whatever you have already in place
- Adding code to existing applications to use managed connections is easy

Managed Connections Summary

- It's a better fit with SQL client-server databases
- It makes your applications behave more sensibly
- During development it’s easy to work with multiple copies of databases
- It makes it easier to deploy to database servers - only one file changes
- It makes it easier to exchange workspaces with SQL data
- A single config file can define connections to *multiple* servers
- A single config file can define connections to *alternative* servers
- Password credentials are automatically and uniquely encrypted in the connection file
- It can be used with embedded SQL
- It's very extendable