

# SQL Managed Connections

---

John Tuohy  
DISD 2018  
March 2018

# The SQL style of accessing data

---

- You connect to a server and database (connect = login)
- Using that connection, you access tables within the database
- When finished you log out of the server
- The server contains the meta-data for the database and its tables
- Portability is achieved by logging into a different server/databases that have similar tables

# The DataFlex style of accessing data

---

- A Filelist stores the locations of all tables
- Each table is opened before using it
- Tables are accessed and updated
- Upon completion tables are closed
- The meta-data for the table is stored in the table itself
- Portability is achieved via relative names in the Filelist, changing Filelists and copying files
  - Developers took great advantage of this portability

# Connectivity – Bridging the difference

---

- An intermediate file (INT file) is defined for each table
- The INT file contained two kinds of information
  1. Server/database connection information for each table
  2. Table meta-data that is not provided by the SQL Server (special column information, index information, relationship information)
- Filelist points to this INT file
- When a table is “opened”
  - It reads the INT file
  - It logs into the database server (if needed)
  - It reads the table meta-data from the database and augments this with information from the INT file
- While this worked it made it difficult to maintain portable, deployable databases (any change required changing or replacing all of the INT files)

# Connection IDs

---

- Connection Ids were created resolve some of these limitations
- The Table's connection information was no longer stored directly in each table's INT file
- Instead it contained an ID that pointed to a server that is defined elsewhere
- Instead of changing connections in each table INT file you changed it in one central location
- The connection server information was stored either
  - in the driver INT file
  - or you wrote the code to do it yourself

# A Table INT connection strings and Ids

- A table INT file with connection string

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

- A table INT file with a connection ID

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

# Connection ID limitations

---

- Where do you define the Connection strings?
  - The driver ini file was too global and too hard to manage
  - The “write your own code approach” was too difficult
- It still used the bottom up table open approach instead of the top down server/database access table approach
- None of our tools used it
- Developers came up with ways to work around this, but it was not easy
- Connections IDs were the right idea, they just didn't go far enough



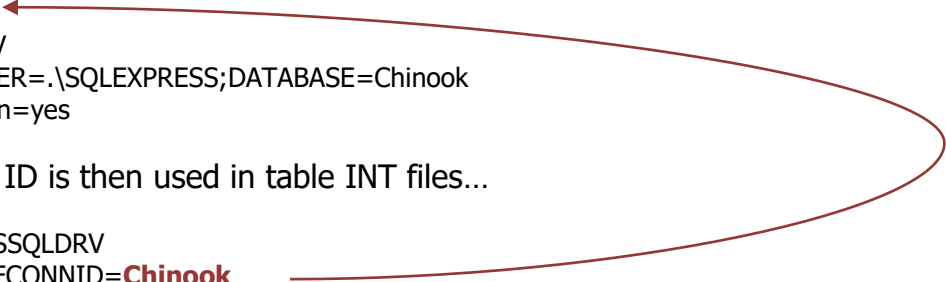
# Introducing Managed Connections

---

# 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"

# Managed Connections

---

- This encourages and even enforces top down access (Login on to server/database then "open" and access tables)
- A single config file can define connections to:
  - multiple servers
  - alternative servers
- Password credentials are automatically and uniquely encrypted in the connection file
- It can be used with embedded SQL
- It provides the basis for switching connections dynamically
- It's very extendable
- It's remarkably backwards compatible

# Studio and Managed Connections

---

- We provide high level tools to configure manage the entire process
  - The Studio, tools and Wizards allow you to add and manage the connections
  - It's easy to create tables using managed connections
    - It's also easy to switch existing SQL table definitions to use managed connections
  - You will see these tools in action today and tomorrow

# Applications and Managed Connections

- The access logic is 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



# 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



# Using Managed Connections

---

- Let's see this in action...



# Supporting Additional Connections

---



# Supporting additional connections

---

- You can define more than one connection in the connections .ini file
  - Alternate connections
  - Multiple connections
- 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
- Multiple connections are defined when your application needs to open tables from multiple servers
  - Each server will have it's own ID
- Let's see this in action...



# 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



# 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



# Encryption and Database Logins

---



# 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 known 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

---

# 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

# Summary

---



# 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
- Adding code to existing applications to use managed connections is really easy