

IBM INFORMIX V.14.10.XC4 - CSDK V.4.50.XC4

v.1a



Agenda

- The new .NET Core provider
- Enhancements to ODBC connection pooling
- Enhanced support for `CLIENT_LABEL`
- Smart trigger support in the Informix Python driver

.Net core

.Net Core

- The CSDK and JDBC connectivity packages provide the libraries for applications to connect to and communicate with the Informix engine
 - Informix has supported a wide range of application interfaces to the engines through these packages
 - C/C++, C#, VB, VS, XA, Tuxedo, Tableau, SQL Linked Server, MS Excel access, .Net provider and many more
- Microsoft has expanded on its .Net platform to provide two different sets of connectivity APIs
 - The .Net provider — Informix has supported this for years
 - The .Net Core — new support in Informix v.14.10.xC4

.Net Core

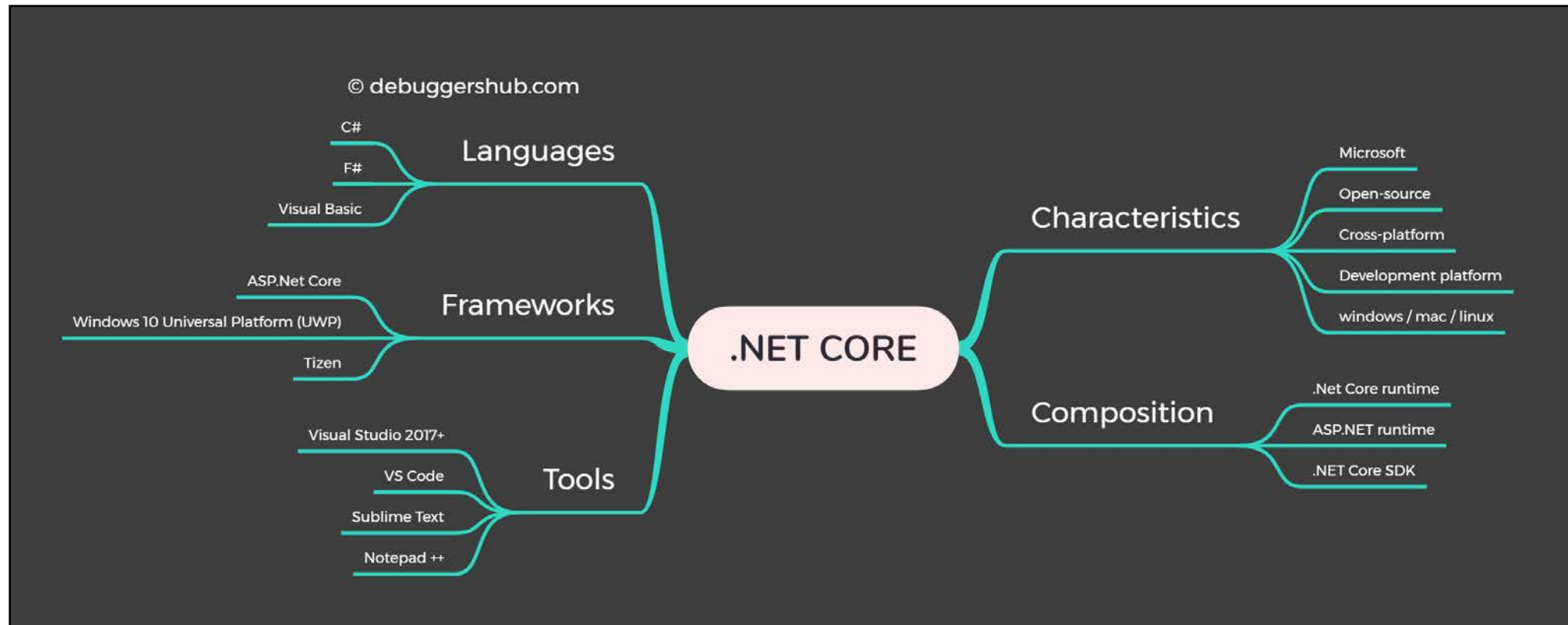
- Wants the difference between .Net provider and .Net Core?
 - According to Microsoft (<https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server>)
 - Both Provider and Core share many components and are, in many cases, code compatible
 - .Net Provider / Framework
 - The legacy driver, used with an existing application infrastructure
 - No need to migrate off, just extend future application functionality by using .Net Core
 - No new development or enhancement is occurring with this product
 - You're using third-party libraries or other libraries not supported by .Net Core
 - Your application platform is not supported by .Net Core
 - For example, some Azure services don't support .Net Core yet
 - Incompatible application O/S
 - .Net core only supports Windows, MacOS and Linux based applications
 - Informix supports .Net Provider v.4x in its most recent CSDK releases

.Net Core

- Wants the difference between .Net provider and .Net Core?
 - According to Microsoft (<https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server>)
 - Both Provider and Core share many components and are, in many cases, code compatible
 - .Net Core
 - Your apps or app services need cross-platform support
 - .Net Core supports Windows, Linux and MacOS
 - You intend to deploy applications and other services in a microservices architecture, such as RH OpenShift or Azure
 - You are “containerizing” your applications or data processing infrastructure using Kubernetes / Docker or other container services
 - .Net provider only works for Windows containers
 - Need greater application performance or scalability than that provided by the .Net provider
 - You need greater flexibility with .Net versions
 - .Net Core supports the installation and simultaneous use of different .Net Core versions
 - Applications can migrate to newer versions as they are ready rather than waiting for all to migrate at once

.Net Core

- Conceptually, this is .Net Core
 - Borrowed from <https://www.debuggershub.com/net-core-the-asp-net-core-architecture/>



.Net Core

- With Informix v.14.10.xC4, the .Net platform is replaced with .Net Core
 - We are using .Net Core v.3.1.1, the current stable release version
 - The v.5 release is still in development and testing
- .Net Core is released on the Windows and Linux CSDK ports now
- The name of the provider, located in `$INFORMIXDIR/bin` is `Informix.Net.Core.dll`
 - This is built on top of the Informix ODBC driver for full compatibility with the Informix engine
- Apparently, the Informix .Net Core provider will be available on [NuGet.org](https://www.nuget.org) in the near future

.Net Core

- For those who want to know the internals of what went into the Informix .Net Core provider, it was created with these packages and libraries:
 - Microsoft Windows Server 2016 Standard — v10.0.14393
 - .NET Core SDK — v3.1.201
 - CMake — v3.13.0-rc3
 - Microsoft Visual Studio Enterprise 2017 — v15.8.6
- The driver was created on Windows for both release platforms
 - It was fully tested on Linux

.Net Core

- With the .Net Core
 - We provide the basic Informix compatible infrastructure
 - You may need to add additional libraries from the .Net runtime or SDK
 - These will come from Microsoft or other providers
- All the existing .Net Framework components (v.4x) will work in .Net Core
 - However you get additional functionality with the new driver

.Net Core

- New functionality
 - Connection pooling
 - Max pool size
 - A global maximum connection value for all connections across all connection pools for an application
 - Range of values between 5 and 2,000
 - You can use this to limit connections to the instance by an application
 - Minimum pool size
 - When a new connection is initialized, the minimum number of connections to the database
 - Range of values between 2 and 1,000
 - For example, if the value is set to 10 and a new connection is made requiring 2 connections, the other 8 are put in an idle state to support future connection requests from other application users
 - `GetIdleConnectionsCount`
 - The ability to get a count, at a global level, of the number of open idle connections
 - `GetActiveConnectionsCount`
 - The ability to get a count, at a global level, of the number of active connections

.Net Core

- New functionality
 - The ability to create and use an ODBC Data Service Name (DSN) in the connection string itself
 - Syntax:
`DSN=my_dsn_name`
- All other .Net functionality, including data type support and tracing, is still available as documented

.Net Core

- An answer to a basic FAQ
 - Can I run a .Net framework application on .Net Core? If so, how?
- Answer: Yes!
 - The names of all the Informix .NET Framework v4.x provider interfaces and methods are the same in the Informix .NET Core provider
 - However you need to
 - Change the assembly name from `IBM.Data.Informix` to `Informix.Net.Core`
 - Recompile the application referencing `Informix.Net.Core.dll` from `$INFORMIXDIR\bin`



ODBC connection pool enhancements

ODBC connection pool

- Starting with the CSDK v.4.50.xC2, the Informix ODBC drivers supports native connection level pooling
- In v.4.50.xC4 the `MinPoolSize` and `MaxConnLimit` functionality is added to support the .Net Core provider
 - Note — if a connection is idle, that connection goes into an “idle” state in the connection pool
 - If the connection is in an idle state for 10 seconds, it is closed and disconnected from the instance

ODBC connection pool

- How do you set these parameters?
 - On Unix
 - In the `odbc.ini` file, in the DSN definition block

```
my_dsn_name
.  
.  
MinPoolSize=2  
MaxConnLimit=1000  
.
```

- Within the application, use the `SQLSetConnectAttr()` function to set the API parameters before attempting to connect to the instance
 - For example:

```
SQLSetConnectAttr(SQL_INFX_ATTR_MIN_CONN_POOL_SIZE=10)  
SQLSetConnectAttr(SQL_INFX_ATTR_MAX_CONN_POOL_SIZE=100)
```

ODBC connection pool

- How do you set these parameters?
 - Within the application by using the connection string

```
DSN=my_dsn_name;MinPoolSize=5;MaxConnLimit=20
```



Client label support in ODBC and SetNet32

Client label support in ODBC/SetNet32

- Client label support was introduced in Informix v.14.10.xC1
 - It allows you to label sessions to track users and / or operations
- A label is created via the `CLIENT_LABEL` functionality
 - Can be set either in the O/S shell as an environment variable or in the application session

- O/S shell or at program invocation

```
export CLIENT_LABEL="My name is Steven, using my_hr_app"
```

```
java myjdbc "jdbc:informix-sqli://myhost:52220:CLIENT_LABEL=jdbc_client1"
```

- Application session


```
set environment CLIENT_LABEL "Susan, using payroll_12"
```

Client label support in ODBC/SetNet32

- To see the labels you can
 - Use the `onstat` utility

```
File Edit View Search Terminal Help
set environment CLIENT_LABEL "Carlton, running a test";

select a.order_num, a.customer_num, b.stock_num, b.quantity
from orders a, items b
where a.order_num = b.order_num
group by order_num, customer_num, stock_num, quantity;
~
```



```
Inst_1: onstat -g env 49

IBM Informix Dynamic Server Version 14.10.FC1B2IE -- 0

Environment for session 49:

Variable                Value [values-list]
CLIENT_LABEL             Carlton, running a test
CLIENT_LOCALE            en_US.8859-1
CLNT_PAM_CAPABLE         1
DBDELIMITER              |
DBPATH                   .
                        [.]
                        [//inst_1]
DBPRINT                  lp -s
DBTEMP                   /tmp
IGNORE_UNDERFLOW         1
INFORMIXDIR              /opt/IBM/informix/14_10
                        [/opt/IBM/informix/14_10]
                        [/usr/informix]
INFORMIXSERVER            inst_1
```

Client label support in ODBC/SetNet32

- To see the labels you can
 - Query the `sysmaster` : `sysenvses` table
 - The `sysenvses` table contains eight data elements which can be viewed by a `onstat -g env` operation
 - For example:

```
envses_sid      51
envses_id       0
envses_name     DBPATH
envses_value    .

envses_sid      51
envses_id       1
envses_name     CLIENT_LOCAL
envses_value    en_US.8859-1

envses_sid      51
envses_id       2
envses_name     NODEFDAC
envses_value    no

envses_sid      51
envses_id       3
envses_name     CLNT_PAM_CAP
envses_value    1

envses_sid      51
envses_id       4
envses_name     DBTEMP
envses_value    /tmp
```

```
envses_sid      51
envses_id       5
envses_name     SHELL
envses_value    /bin/bash

envses_sid      51
envses_id       6
envses_name     SUBQCACHESZ
envses_value    10

envses_sid      51
envses_id       7
envses_name     PATH
envses_value    ./opt/IBM/informix/14_10/bin:/usr/local/sbin:/usr/sbin:/home/te

envses_sid      51
envses_id       8
envses_name     CLIENT_LABEL
envses_value    Carlton, still testing this
```

Client label support in ODBC/SetNet32

- The new client label is the eighth element
 - With this information, you can “tag” an application, use the tag to identify the sessions from that app, then investigate their operations

```
----- stores@inst_1 -  
  
select * from sysmaster:sysenvses  
where envses_id = 8
```

```
envses_sid      51  
envses_id       8  
envses_name     CLIENT_LABEL  
envses_value    Carlton, still testing this  
  
envses_sid      49  
envses_id       8  
envses_name     CLIENT_LABEL  
envses_value    Carlton, running a test
```


Client label support in ODBC/SetNet32

- With v.4.50.xC4, you can also set a label through the ODBC driver and the SetNet32 application
 - As part of the connection string

```
DSN=my_dsn_name;CLIENT_LABEL=my_label
```

- Through the the `SQLSetConnectAttr()` function

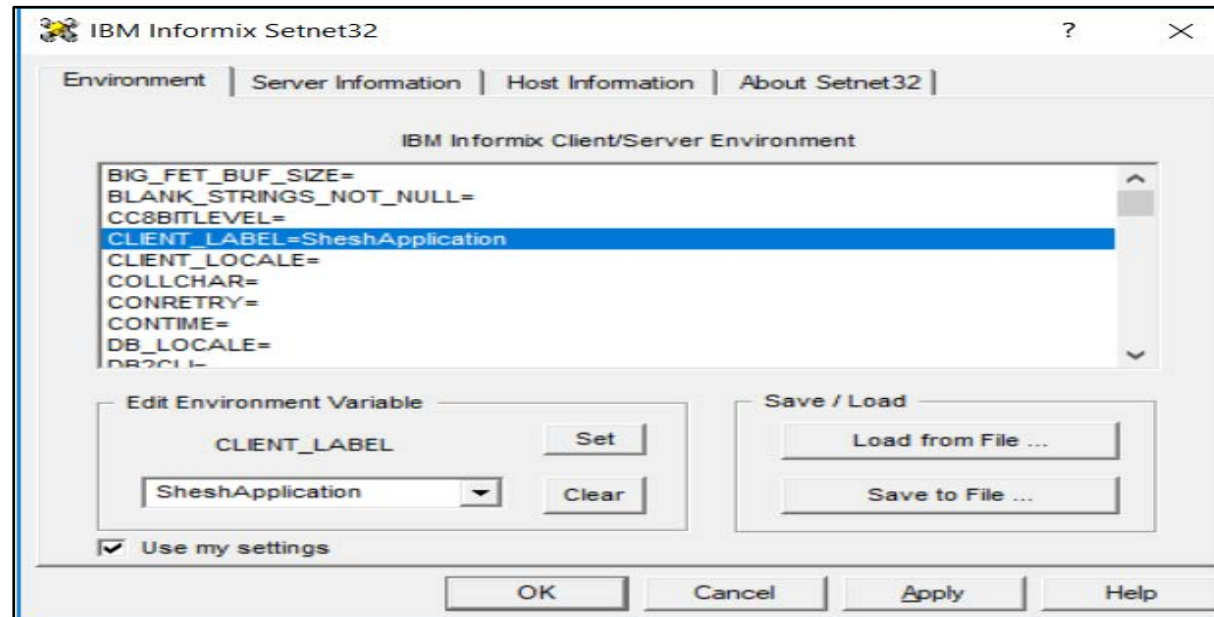
```
SQLSetConnectAttr(SQL_INFX_ATTR_CLIENT_LABEL=my_label)
```

- In the `odbc.ini` — Unix / Linux only!

```
my_dsn_name
.
CLIENT_LABEL=my_label
.
```

Client label support in ODBC/SetNet32

- Setting the client label
 - Using the SetNet32 application and the CLIENT_LABEL variable





Smart trigger feature in the Informix Python driver

Smart trigger support

- Smart triggers were introduced in Informix v.12.10.xC9
 - Was created to work around conditions where an application must constantly ping the instance to see if specific data conditions exist so the application can do some work
 - For example, was data inserted into a table that must be picked up and sent to another application or target
 - Using Java and JDBC, you can create a “smart trigger” that monitor changes to data and when a “triggering condition” occurs, send an alert and the data to the application to work on
- In Informix 12.10.xC10 additional enhancements were made including support for receiving triggering events after the application reconnects to the instance

Smart trigger support

- In Informix v.14.10.xC4, smart trigger support is expanded to include support for Python applications
 - 6 new functions were added to support this functionality
 - `IfxPy.open_smart_trigger()` - open a smart trigger session
 - `IfxPy.get_smart_trigger_session_id` - get the session IDs of existing smart trigger sessions
 - `IfxPy.join_smart_trigger_session` - join an existing smart trigger session ID
 - `IfxPy.register_smart_trigger_loop` - open a smart trigger session with a loop handled by Informix Python driver
 - The session will wait and loop inside the Python driver
 - `IfxPy.register_smart_trigger_no_loop` - open a smart trigger session without a loop handled by Informix Python application
 - The session will wait and loop inside the application
 - `IfxPy.delete_smart_trigger_session` - delete a specific smart trigger session

Smart trigger support

- Please note, this functionality is NOT in the general release of the product
 - Customers must make a special request to receive it

Questions

