Integration

Information Management software

Featured integration solutions for IMS Version 13

- **IBM IMS Enterprise Suite**
- IBM Cognos Business Intelligence (BI) 10.2
  - offers a data connection type for direct connections to IMS databases.
- IBM InfoSphere Data Explorer 9.0
  - directly access to IMS operational data to explore critical business data.
- WebSphere DataPower® Integration Appliance V6.0 (5725-K52)
  - IMS Synchronous Callout Support
  - Access to IMS transactions that are running in IMS TM.
  - Access to databases in IMS DB

03- IMS13 Integration: 152

IMS TM Resource Adapter Version 13

- Based on version 1.5 of the JEE Connector Architecture
- Documented in IMS Version 13 Application Programming (SC19-3646-00)
- Provides all key features available in Version 12:
  - Support for RACF password phrases from 9 -100 characters
    - APAR PM91898 (TM RA V13.2.0):
      Note: IMS Connect V13 PM91312 and IMS V13 PM85849 are required
  - Support for changing RACF passwords from Java applications
  - Support for RACROUTE VERIFY return codes
  - Support for IBM WebSphere Application Server Version 8 and its resource workload routing function
  - Support for both synchronous and asynchronous callout requests from IMS applications
  - Support for integrating IMS MFS services in business processes through IBM Integration Designer and IBM Process Server
  - Support for a single message-driven bean (MDB) to pull callout messages from more than one IMS data store

If the password phrase is used, an internal PING message will be first sent to IMS Connect (ICON) to make sure that the level of
IMS Connect does support the password phrase. Otherwise, the regular 8
bytes password value will be used for verification.

Note: IMS Connect V13 PM91312 and IMS V13 PM85849 are required for this function to work properly. Both APARs can be applied independently to TMRA and IMS Connect, however, both APARs must be applied to use the password phrase feature

Support for multiple data stores per IMS activation specification for callout messages enables a single message-driven bean (MDB) to pull callout messages from more than one IMS data store. A shared-queues environment removes need to duplicate the MDB application to connect to each IMS member in the shared queue.

DLIModel

- IMS Enterprise suite for z/OS , V2.1 is the last release to provide the DLIModel Utility plug-in.
- Explorer provides ability to import DLIModel projects
  - Explorer does not support XML DB or DB Web Services
- Customers using IMS Database Web Services should transition to using the IBM Data Studio which leverages the IMS Universal Drivers

Customers using DLIModel Utility support for Database Web Services should transition to using the IBM Data Studio Database Web Services support, which leverages the IMS Universal Drivers.

Rational Developer for System z

- Rational Developer for System z IMS Edition
  - No support for IMS ES 2.2 Explorer for Development
  - Move to RDz 90 day trail
- Rational Developer for System z 8.5 and later
  - RDz 90 day trial download site
    - **http://www.ibm.com/developerworks/downloads/r/rdz/**

03- IMS13 Integration: 155

Integration Enhancements

- Concurrent Application Threads Enhancement
- IMS SQL Support
- IMS Enterprise Suite 3.1
  - IBM IMS Data Provider for Microsoft .NET
  - SOAP Gateway Updates
  - Connect API for Java V2.2 Updates
  - Explorer for Development Updates
  - Java Message Service (JMS) API
    - IMS 13 synchronous program switch support
- IBM IMS Explorer for Administration
  - Extension to the IBM Tools Base v1.4 Administration Console component

03- IMS13 Integration: 156

Concurrent Application Threads Enhancement

- Partition Specification Table (PST) used for
  - Active dependent regions (MSG/BMP/IFP/JMP/JBP)
  - CICS/DBCTL threads
  - Open Database Access threads
- Customers continue to require more PSTs!
  - 31 dependent regions - 1980 (IMS 1.1.6)
  - 999 dependent regions - 1995 (IMS 5.1)
  - 4095 dependent regions - 2013 (IMS 13)
- Related parameters
  - MAXPST=
  - PST=
  - MAXTHRDS= for ODBM
  - MAXTHRDS=, MINTHRDS= for DBCTL
- Details in IMS13 System Sections

This slide is a reminder as IMS integration solutions grow the need for more application threads (PSTs) will increase.. Details are covered in IMS 13 systems section

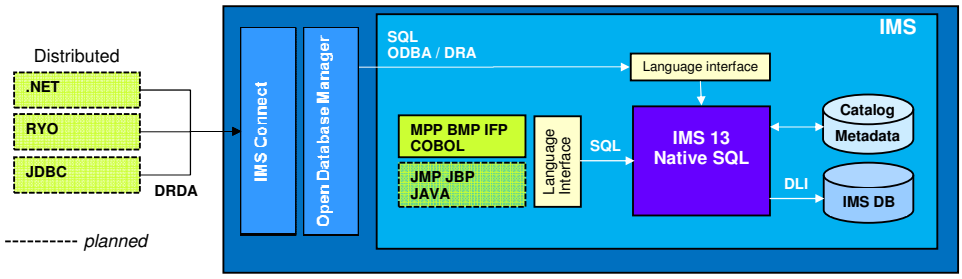
ODBM MINTHRD Default 1 for RRS=N

- Problem
  - EXCESSIVE TCB ATTACH AND DETACH PROCESSING IN AN ODBM RRS=N ENVIRONMENT.
- Solution
  - MINTHRD value in an RRS=N ODBM environment is set to approximately 62% of the value of MAXTHRD
  - IMS 12 PM63976 UK82608
  - IMS 11 PM63977 UK82609
- Benefit
  - performance issues associated with attaching and detaching threads are reduced.

Improved performance by changing the MINTHRD default from 1 to 62% of the MAXTHRD value to minimize the attach/detach processing.

IMS SQL Support

- Native SQL for COBOL and distributed applications (.NET/JDBC)
  - COBOL Details in IMS13 Application Programming Enhancements Section
- Provides standard SQL keywords to easily access IMS data
  - ✓ SELECT, INSERT, UPDATE, DELETE
  - ✓ Uses Dynamic SQL programming model
  - ✓ Converts SQL statements to DLI calls
  - ✓ Supports a subset of SQL keywords that are currently supported by IMS Universal JDBC driver
- Uses database metadata in IMS Catalog
  - ✓ No need to generate metadata for use in applications

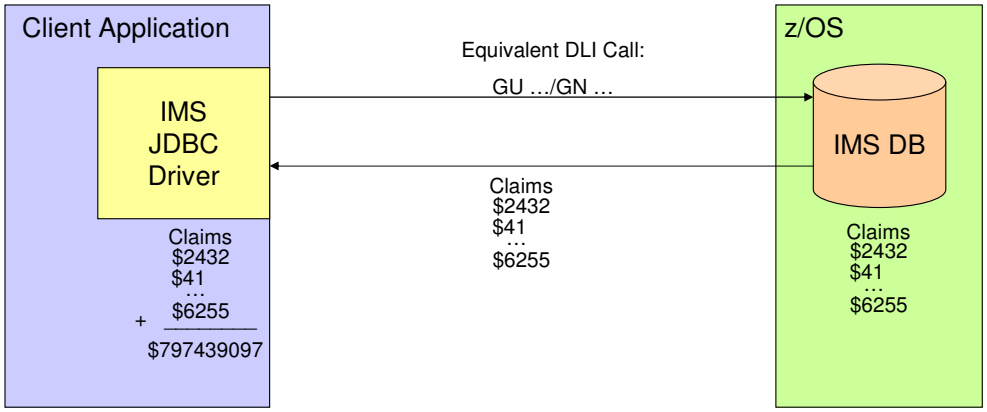


IMS DRDA DDM command support for native SQL enhancement

- The DDM command support for native SQL requires the Open Database Manager (ODBM) component of the IMS Common Service Layer (CSL).
  - ODBM translates the DDM commands into SQL and then routes the SQL calls to the appropriate IMS system.
  - The receiving IMS system's native SQL translates the SQL into DL/I.
- IMS Data Provider for Microsoft .NET will use this support
- IMS Universal Drivers to be updated via service process
  - Enables SQL processing to be handled directly by IMS instead of on the client side,
  - Results in increased performance for the IMS Open Database solution.

SQL Parsing in IMS Java Code

- **Requirement** - Improve performance for SQL data aggregation
- Example
  - How much money has my insurance company paid out in claims for the year 2011?
  - SELECT SUM(CLAIMAMOUNT) FROM CLAIMS WHERE YEAR=2011

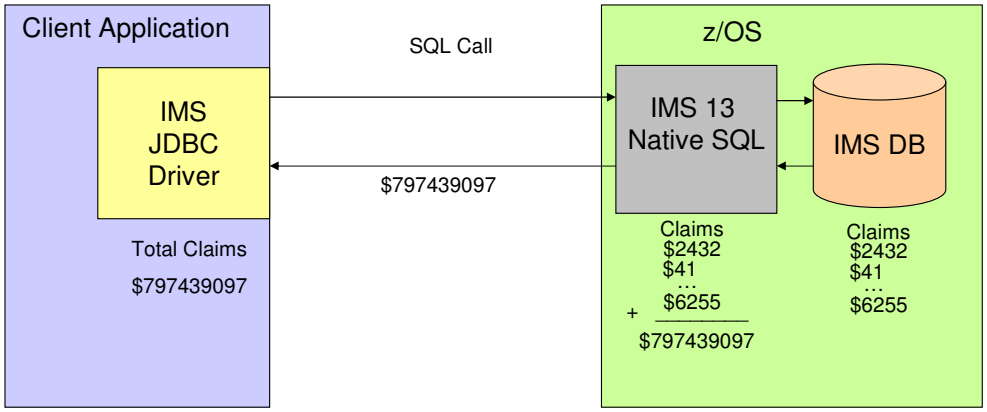


For the sake of simplicity to demonstrate the use case, some intermediate components like ODBM and IMS Connect are not shown in this chart. For a full view look at the underlying architecture chart.

In the scenario above a very simple summation call could result in having a large amount of data being transferred over the network. In this case all of the claims data from the year 2011 is being streamed to the client side

SQL Parsing in IMS

- Solution IMS Java intends to use IMS 13 Native SQL
  - IMS Service Process
- Example with IMS SQL call handler
  - How much money has my insurance company paid out in claims for the year 2011?
  - SELECT SUM(CLAIMAMOUNT) FROM CLAIMS WHERE YEAR=2011



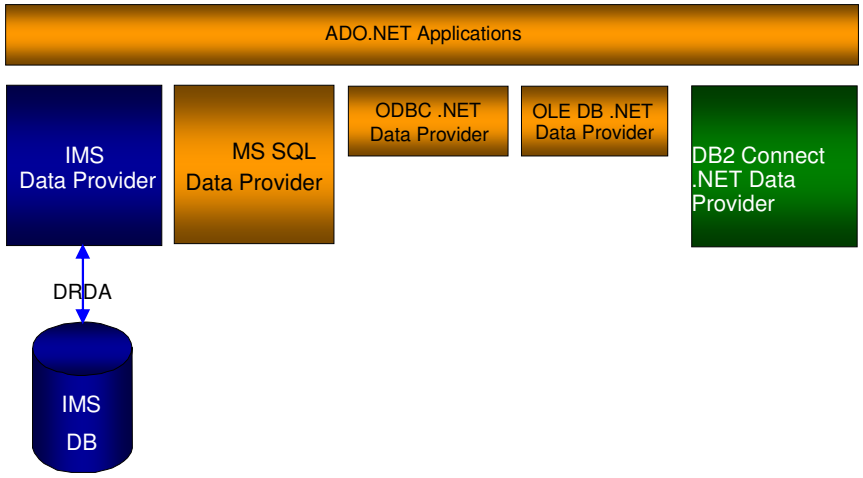
In this picture that Native SQL engine now handles the data aggregation on z/OS and only the final summation is streamed over to the JDBC driver.

IBM IMS Data Provider for Microsoft .NET

- Available via the IMS Enterprise Suite V3.1 service process
  - Based on the .NET Version 4.0 specifications
  - Application developers can use their preferred development environment, such as Microsoft Visual Studio, to call the IMS provided APIs.
  - Support is for IMS 13 or later
- Benefit
  - Provides the solution for Microsoft .NET-based applications to access and manipulate IMS data

IMS Data Provider for Microsoft .NET simplifies development of Microsoft .NET applications that are written in C#, Visual Basic, and other ADO.NET-compliant languages to access IMS data

IBM IMS Data Provider for Microsoft .NET Architecture



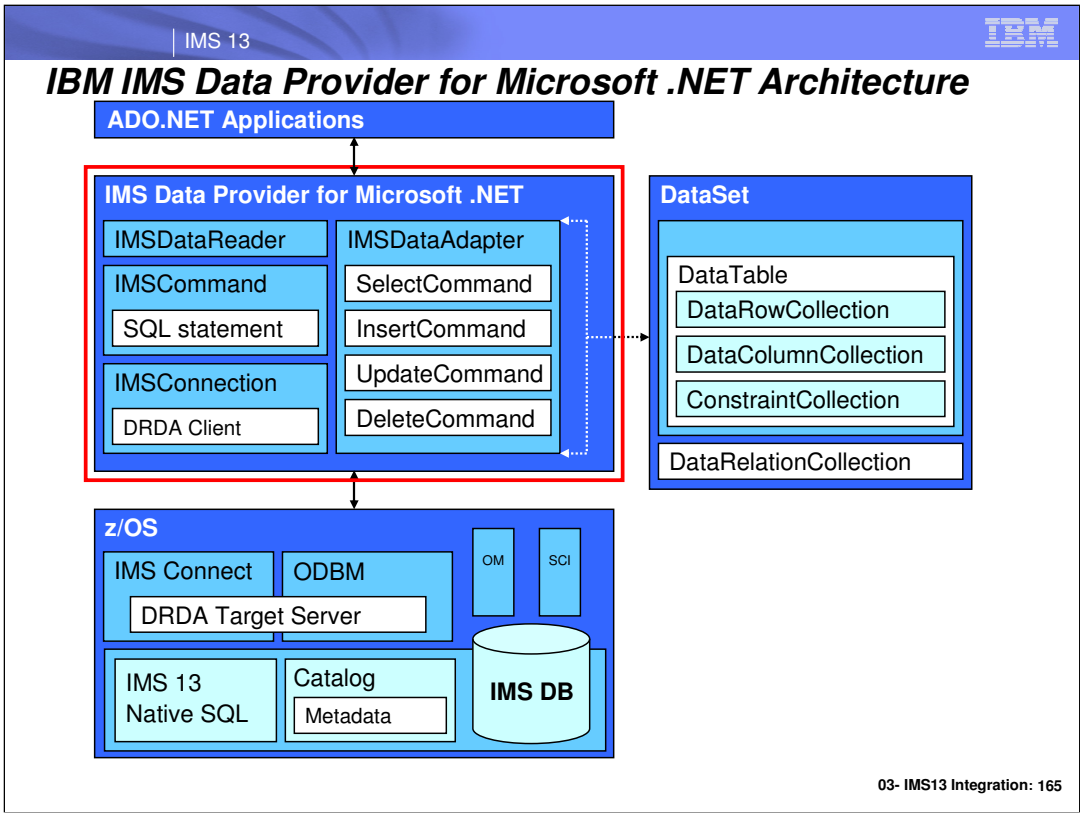
The .NET Framework from Microsoft is the building blocks to build applications by using the class library in the framework, supporting several programming languages that allows language interoperability. Programs written for the .NET Framework run in the Common Language Runtime (CLR) runtime environment, an application virtual machine that provides crucial services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

ActiveX Data Object for .NET, or ADO.NET, is a set of software components for accessing data and data services. ADO.NET is part of the base class library that is included with the Microsoft .NET Framework.

.NET data providers are software components that enable an ADO.NET consumer to interact with a data source. The .NET Framework includes the System.Data.Common namespace, which provides a set of base classes that can be shared by any .NET data provider. This namespace facilitates a generic ADO.NET database application development approach with a consistent programming interface.

MS SQL is accessed directly via ADO.NET API. For 3rd party DMBS and other Data Access standards Data Providers are needed.

.NET comes with Data Providers for ODBC and OLE out of the box. IMS is added to the list of supported DBs by implementing .NET Data Provider.



Red box – contains all required ADO.NET interfaces (DataReader, DataAdapter, Command, Connection). Data retrieved from IMS will be stored in a standard DataSet class (in-memory datastore) or IMSDataReader which is similar to ResultSet in Java.

The .NET DP will be a DLL that .NET apps will be using directly. The DLL connects to z/OS via TCP/IP and DRDA protocol (just like IMS Type 4 Universal Drivers).

IMS Connect and ODBM address spaces are required and together they form DRDA Target Server.

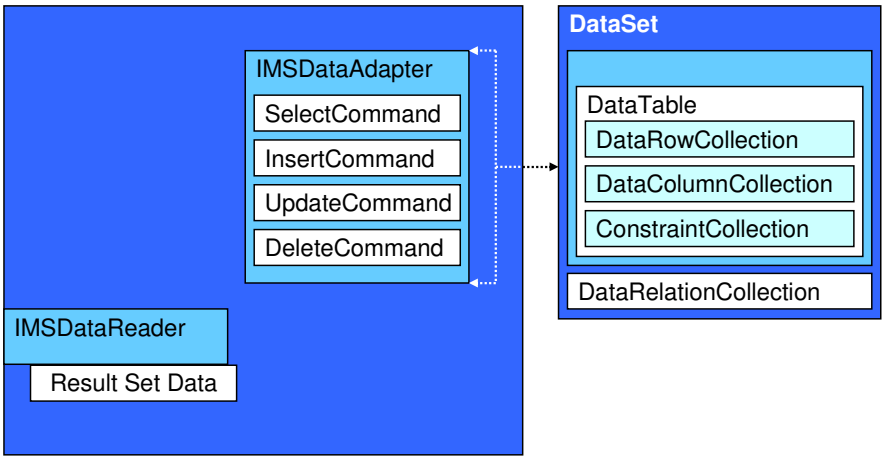
In IMS, Native SQL takes care of processing SQL queries and sending results back. Catalog feature is used for metadata, so no offline metadata is necessary (in fact, it will not be supported).

Both online and offline (disconnected) modes of operation will be supported.

IMSConnection class includes the properties and methods that is required to establish a connection with the IMS DRDA Server

IMSCommand class represents an SQL statement to execute against a data source by specifing what type of SQL interaction you want to perform with a IMS database. This object together with the IMSConnection object provides the IMS .NET Data Provider customers with a connected data approach.

IBM IMS Data Provider for Microsoft .NET Architecture



DataSet a memory-resident relational representation of IMS data managed by IMSDataAdapter

DataReader object is used for fast-forward reading streams of IMS data

Data retrieved from IMS will be stored in a standard DataSet class for disconnect mode processing and DataReader for connect mode processing.

IMSDataReader serves similar purpose as a DataReader in the ADO.NET technology. This object is used for fast-forward reading streams of IMS data. This object cannot be used for writing data. Due to the stream behavior, once some data is read, you must save it for your purpose since you will not be able to go back and read it again.

IMSDataAdapter similar to the DataAdaptor object in ADO.NET manages connection and interaction with IMS and gives the users of the IMS .NET Data Provider a disconnected behavior.

This object opens a connection only when required and closes it as soon as it has performed the intended task.

Dataset is an in-memory data store and the user keeps manipulating the dataset till they are ready to push the change back to IMS using the IMSDataAdaptor. Note that the IMSConnection object needs to be instantiated before calling the IMSdataAdaptor object, but not opened. The IMSDataAdapter will open and close the connection during Fill and Update method calls transparently to the users.

Connecting to IMS databases

- IMSConnection Class
  - Server=myserver, - IMS Connect Host name/Network address
  - 5555; - IMS Connect Default DRDA Port Number
  - Database=DFSSAM09; - PSB name in IMS Catalog
  - Datastore=IMS13A; - IMS name defined to ODBM
  - UID=imsadm; - SAF UserID
  - PWD=ab1d; - SAF Password
  - Connect Timeout=30"; - time to wait for establishing a connection

03- IMS13 Integration: 167

Connect and Disconnect Mode Concepts

- Connected architecture
  - Every select, insert, update, delete is an access to the IMS database
- Disconnected architecture
  - Fetch the data from IMS database into local DataSet
  - Manipulate data w/o accessing the IMS database
  - When complete commit all your changes to the IMS database.

In connected architecture for each data add, retrieve, update, and delete operation requires access to the database. such as for every select, insert, delete ,update your application will access the database.

In disconnected architecture once you fetch the data you can perform operations to the data without accessing the database. and when you have completed all the data operations then you commit all your changes to database.

Connect mode

1. IMSConnection Open connection

   Obtain socket connection from connection pool

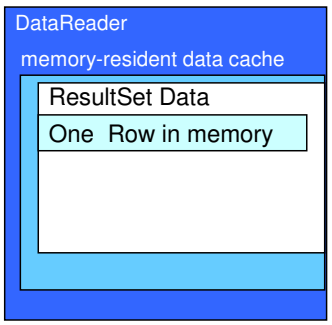
2. IMSCommand build SQL string

3. IMSDataReader retrieve data from IMS store into DataReader
   - First SQL call allocates a program specification block (PSB)
   - Database locks held until Close connection

4. IMSConnection Close connection

   Deallocates the PSB

   Return socket connection to connection pool



Use DataReader when:

- Dealing with large volumes of data—too much to maintain in a single cache.
- Reduce the memory footprint of your application.
- Want to avoid the object creation overhead associated with the DataSet
- Want to perform data binding with a control that supports a data source that implements IEnumerable
- Wish to streamline and optimize your data access
- Reading rows containing binary large object (BLOB) columns

Note each Open, SQL call and Close Connection causes an allocate/deallocate of the PSB

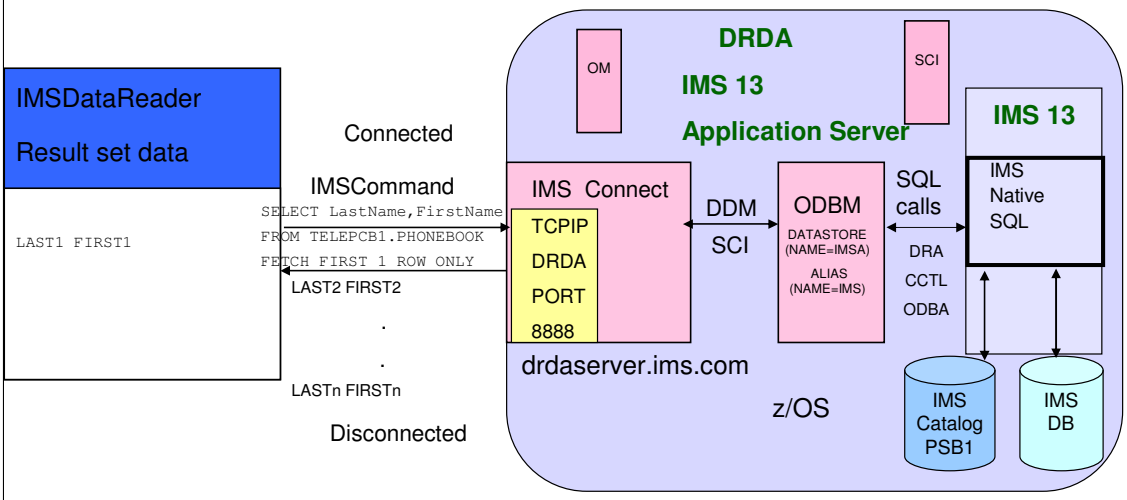
C# Application Connect mode Read only

Example.cs

```
using IBM.Data.IMS;

static void IMSReader()
{
        // Use connection string to configure connection properties
        IMSConnection connection = new IMSConnection("Data source = MyIMS,5555;
           Database = Insurance");
        // Establish connection to IMS database
        connection.Open();

        // Specify SQL query in the IMSCommand object
        IMSCommand command = new IMSCommand("SELECT * FROM PCB.CUSTOMERS",
                     connection);

        // Execute query via the DataReader object
        IMSDataReader reader = command.ExecuteReader();

        // Iterate through results and output on the screen
        while (reader.Read())
           Console.WriteLine(reader.GetString(0));

        // Close the reader
        reader.Close();

        // Close the connection
        connection.Close();
}
```


Point of example is to show how IMSDataReader is used for read only processing while in a connection

IMSDataReader



An IMSDataReader implements a DataReader in the ADO.NET technology. This object is used for fast-forward reading streams of data.

This object cannot be used for writing data. Due to the stream behavior, once some data is read, you must save it for your purpose since you will not be able to go back and read it again.

```
SELECT LastName,FirstName FROM TELEPCB1.PHONEBOOK
FETCH FIRST 1 ROW ONLY
```

C# Application Connect Mode change data

Example.cs

```
using IBM.Data.IMS;

static void IMSWriter()
{
        // Use connection string to configure connection properties
        IMSConnection connection = new IMSConnection("Data source = MyIMS,5555;
           Database=Insurance");
        // Establish connection to IMS database
        connection.Open();

        // Specify SQL command in the IMSCommand object
        IMSCommand command = new IMSCommand("INSERT INTO PCB.CUSTOMER (NAME,
      POLICY) VALUES ('EVGENI', 1210050000)", connection);

        // Execute command
        int i = command.ExecuteNonQuery();

        // Close the connection
        connection.Close();
}
```

- INSERT, UPDATE and DELETE commands are used identically

Point of example is to show how IMSCommand is used for SQL commands that can change the IMS data while in a connection

Disconnect mode

- **DataSet -** Disconnected memory-resident data cache
  - Connections are not maintained for long periods
  - Database locking does not occur

1. IMSConnection set connection parms
2. IMSCommand build SQL string
3. IMSDataAdapter

   Open connection

   Retrieve data from IMS store into DataSet

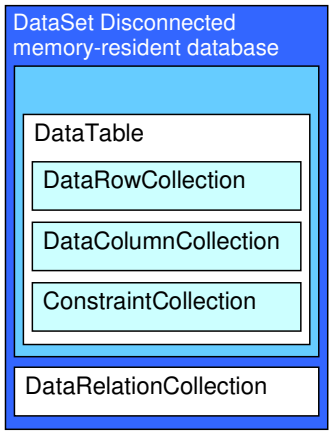
   Allocate PSB

   Close connection

   Deallocate PSB
4. Change data in DataSet
   1. IMSDataAdapter FILL/Update Methods

      Open connection
      Allocate PSB
      Write changes from DataSet to IMS
   2. Close connection

      Deallocate PSB



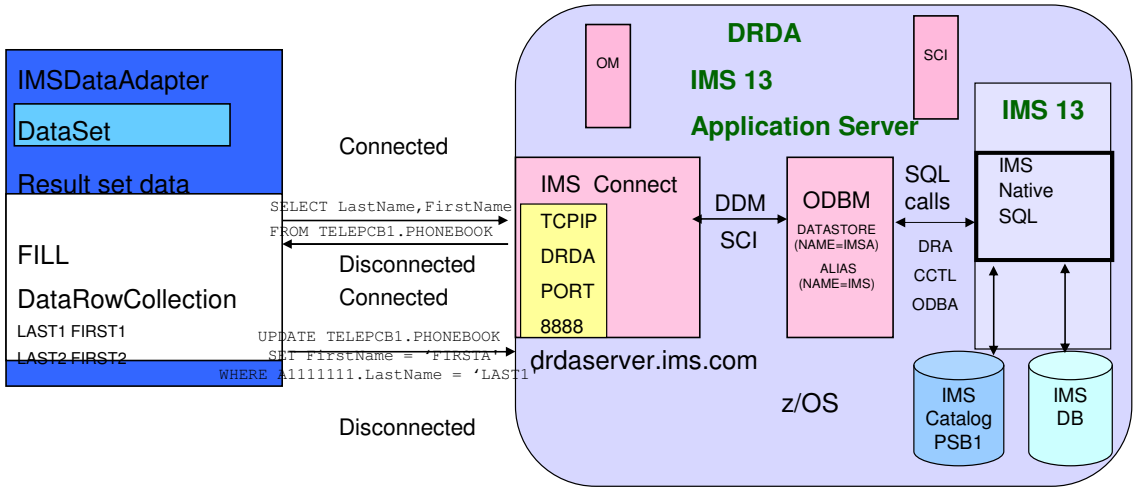
Use DataSet when:

You require a disconnected memory-resident cache of data, so that you can pass it to another component or tier within your application.

You are working with data retrieved from multiple data sources, such as multiple databases, tables, or files.

You want to perform data binding against a control that requires a data source that supports List.

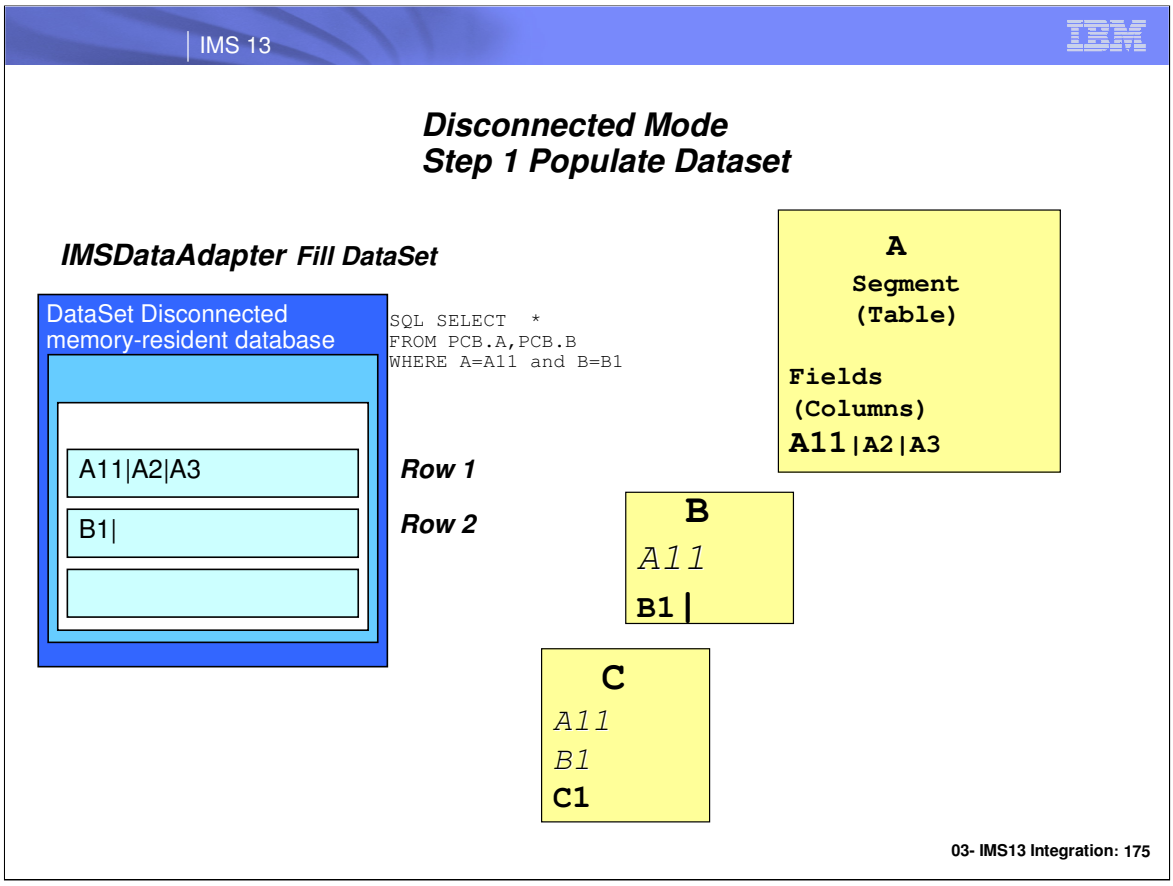
IMSDataAdapter



03- IMS13 Integration: 174

Similar to the DataAdaptor object in ADO.NET, the IMSDataAdapter manages connections and interactions with IMS and gives the users of the IMS .NET Data Provider a disconnected behavior. This object opens a connection only when required and closes it as soon as it has performed the intended task. Here the SQL statement is held in its properties and when it is executed the result set is filled into the corresponding DataSet specified by the application.

The Dataset is an in-memory data store and the user keeps manipulating the dataset till they are ready to push the change back to IMS using the IMSDataAdaptor. Note that the IMSConnection object needs to be instantiated before calling the IMSdataAdaptor object, but not opened. The IMSDataAdapter will open and close the connection during Fill and Update method calls transparently to the users.

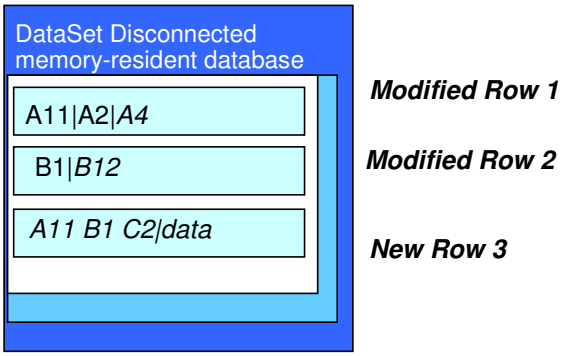


This slide provides a mapping of IMS hierarchical database concepts and relational database concepts. It also shows how IMS Foreign key is used to provide referential integrity. This is used to enforce ADO.NET fill/update processing.

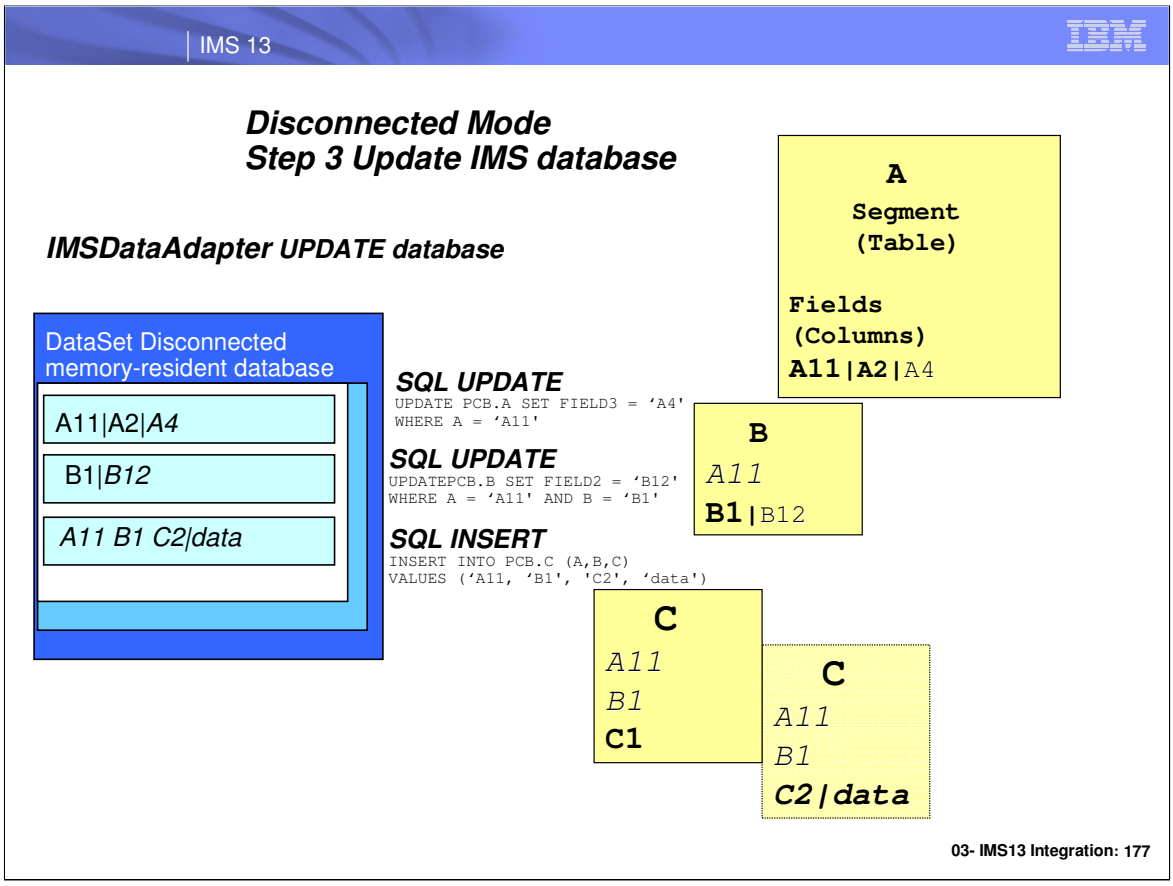
Fill your **DataSet** with current data by using the primary key values of the rows returned by the **SelectCommand**

Disconnected Mode
Step 2 change data in Data Set

Change DataSet



Make changes to DataSet while still disconnected from IMS



Any row in the returned result set whose primary key corresponds to an existing row in the DataSet will be used to update that row, and that row's state will always become DataRowState.Modified, *even if the returned row is identical to the current row*

Any row in the returned result set whose primary key doesn't correspond to any existing row will be used to create a new row, and that row's state will become DataRowState.Added

Any row in the DataSet that doesn't correspond to a row in the returned result set will stay at DataRowState.Unchanged

For this example, the Update method executes a two UPDATE statements, followed by an INSERT statement due to the ordering of the rows in the DataSet.

UPDATE PCB.A SET FIELD3 = 'A4' WHERE A = 'A11'

UPDATE PCB.B SET FIELD2 = 'B12' WHERE A = 'A11' AND B = 'B1'

INSERT INTO PCB.C (A,B,C) VALUES ('A11, 'B1', 'C2', 'data')

Example of Using Disconnected Mode

Example.cs

```
IMSConnection conn = new IMSConnection("Data source=myims,5555;Database=Insurance");
string queryString = "SELECT * FROM PCB.CUSTOMERS";

//Create new DataAdapter object
IMSDataAdapter adapter = new IMSDataAdapter(queryString, conn);

//Create CommandBuilder object for automatic command generation
IMSCommandBuilder builder = new IMSCommandBuilder(adapter);
DataSet customers = new DataSet();

// This call will execute IMSDataReader to fill DataSet with query results
adapter.Fill(patients, "Patients");

foreach (DataRow pRow in customers.Tables[“Customers"].Rows)
    Console.WriteLine(pRow["NAME"]);

//Change value in Dataset
DataRow myRow = customers.Tables[“Customers"].Rows[1];
myRow[“CUSTNUM"] = 25;

//Let CommandBuilder generate Update query
builder.GetUpdateCommand();

//Flush the changes back to IMS
adapter.Update(customers, “Customers");

//Clear existing Dataset and refill it with updated data
customers.Clear();
adapter.Fill(customers, “Customers");
```


03- IMS13 Integration: 178

Generic Coding

- .NET Framework outlines the "**generic coding**", or "**factory-based**" interface that is supported by IMS Data Provider
  - Facilitates generic ADO.NET application development, constant programming interface across different databases
  - When using this technique, proprietary class names, such as IMSConnection, are replaced with common names, such as DbConnection

Example.cs

```
using IBM.Data.IMS;

static void GenericReader()
{
     DbProviderFactory factory = DbProviderFactories.GetFactory("IBM.Data.IMS");
     DbConnection connection = factory.CreateConnection();
     DbConnectionStringBuilder sb = factory.CreateConnectionStringBuilder();

     connection.ConnectionString = sb.ConnectionString;
     connection.Open();

     DbCommand command = new DbCommand("SELECT * FROM PCB.CUSTOMERS", connection);
     DbDataReader reader = command.ExecuteReader();
```


To write generic code that either is not tied to a particular database or supports several different databases, the .NET Framework provides a factory-based interface that is supported by the IMS™ Data Provider for Microsoft .NET.

The .NET Framework features a namespace that is called System.Data.Common, which includes a set of base classes that can be shared by any .NET data provider. This namespace facilitates a generic ADO.NET database application development approach, offers a constant programming interface across different databases, and enables the factory design model for client database applications. These features increase design flexibility and reduce module maintenance cost.

When you use this technique, proprietary class names such as IMSConnection are replaced with common names, such as DbConnection.

Transactions

- IMS Data provider supports Local **Transactions**
  - IMSTransaction object is responsible for rolling back and committing database transactions

Example.cs

```
IMSCommand command = connection.CreateCommand();
IMSTransaction transaction;

// Start a local transaction.
transaction = connection.BeginTransaction("SampleTransaction");
command.Transaction = transaction;

try
{
    command.CommandText = "INSERT INTO REGION (ID, NAME) VALUES (100, 'Portland')";
    command.ExecuteNonQuery();

    command.CommandText = "INSERT INTO REGION (ID, NAME) VALUES (200, 'Vegas')";
    command.ExecuteNonQuery();

    transaction.Commit(); // Attempt to commit the transaction.
}
catch (Exception ex)
{
    transaction.Rollback(); // Attempt to roll back the transaction.
}
```

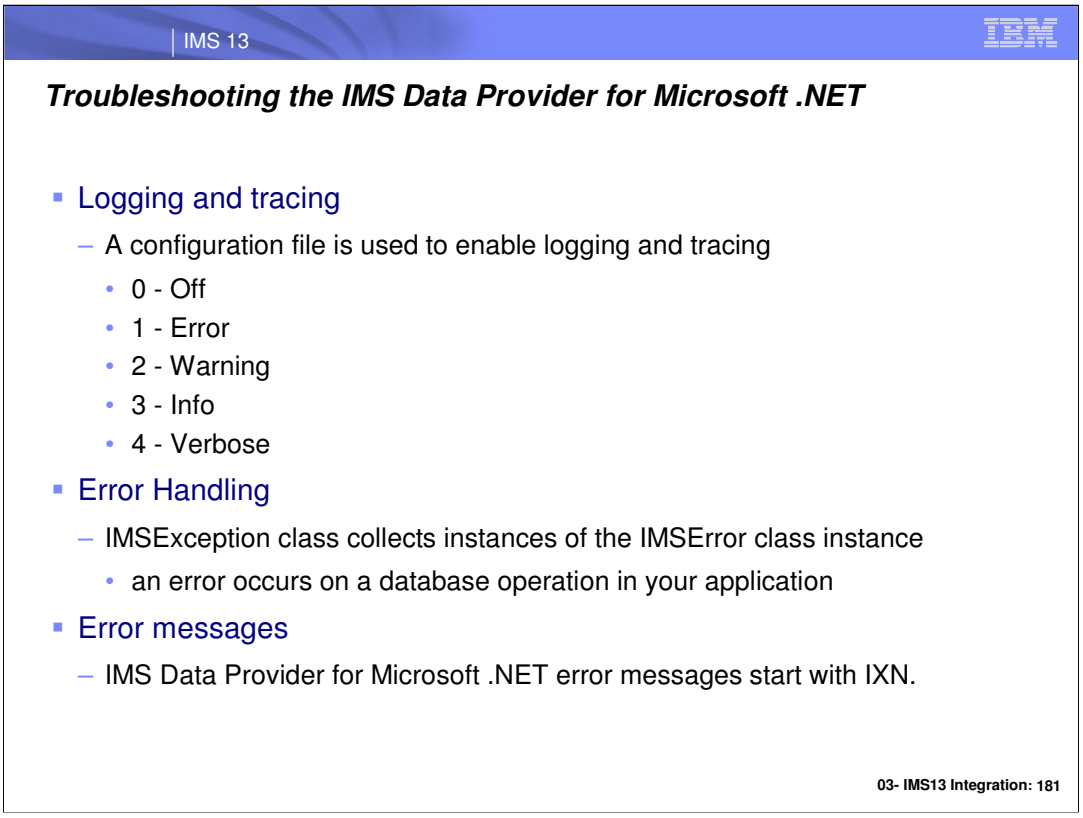

By default, every SQL command is autocommitted.

If multiple SQL statements need to be executed as a single transaction, the Transaction property of the IMSCommand object must be initialized to an IMSTransaction object. An IMSTransaction object is responsible for rolling back and committing database transactions. When the application creates an IMSTransaction object by calling the BeginTransaction() method on the IMSConnection object. All subsequent operations associated with the transaction (for example, committing or aborting the

transaction), are performed on the IMSTransaction object.

Note when using IMSTransaction CLOSE will rollback any pending transactions:

```
// Close the connection
connection.Close();
```



Error messages and logging are available to facilitate troubleshooting.

An instance of the IMSError class is created whenever an error occurs on a database operation in your application. Each instance of IMSError created by the IMSDataAdapter is managed by the IMSErrorCollection class, which in turn is created by the IMSException class.

Error handling

IMS Data Provider for Microsoft .NET provides an IMSException class that collects instances of the IMSError class. Catching exceptions in your code can prevent the application from failing and provide a relevant error message to your user.

Logging and tracing

You can enable logging and tracing by providing a configuration file and specifying the trace level.

Error messages for IMS Data Provider for Microsoft .NET

Error messages for the IMS Data Provider for Microsoft .NET starts with IXN. Some error messages that are related to connections are followed by errors from the DRDA server. Errors that are related to SQL queries often include an error code from the SQL support in IMS.

Security

- Security Considerations
  - IMS Data Provider for Microsoft .NET supports Secure Sockets Layer (SSL) protocol when the IMS host system uses the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS).
  - IMS managed Userid and PSB security check
    - Resource access security (RAS) or APSB
      - A security check is performed by RACF to determine if the user is authorized to use the PSB

03- IMS13 Integration: 182

Restrictions

- The IMS Data Provider for Microsoft .NET has the following restrictions:
  - Stored procedures currently are not supported.
  - SQL queries must be issued directly from .NET applications.
  - Global transactions are not supported.
  - The command builder class that is used for automatic generation of commands in disconnected mode is not implemented
  - Not all SQL statements are supported.
    - see the ***IMS Version 13 Application Programming APIs*** information.
  - Not all IMS data types are supported in the current release.
    - For the list of supported IMS data types,
      - ***IMS Enterprise Suite > IMS Enterprise Suite Version 3.1 > IMS Data Provider for Microsoft .NET > Developing .NET applications with the IMS Data Provider for Microsoft .NET***

In the 1st release, .NET data provider only supports local transactions (single participant).

System Requirements

- **Software requirements**
  - IMS Version 13, with the following required functions and APARs:
    - IMS Connect
    - CSL - Open Database Manager (ODBM) , OM and SCI
    - IMS catalog must be properly configured.
    - IMS Version 13 APARs PM96324 and PI05437 must be applied.
  - Microsoft .NET Framework Version 4.0 or later
  - Microsoft Visual Studio 2010 or later
- Supported Platforms
  - Windows XP systems with Service Pack 2 or later (32- or 64-bit)
  - 1.6GHz or faster processor
  - 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512 MB if running in a virtual machine)
  - 3 GB of available hard disk space

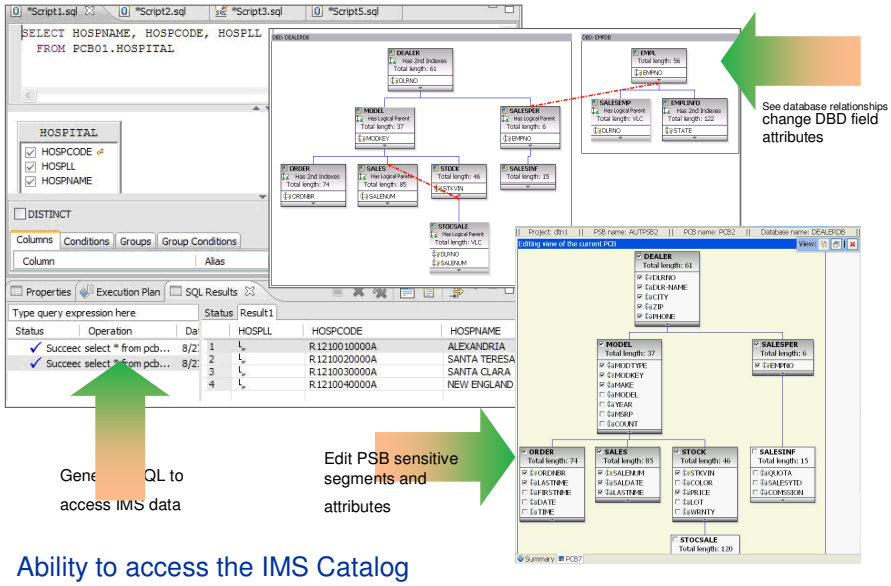
IMS Data Provider for Microsoft .NET installation

- Go to the **IMS Enterprise Suite download site** and log in.
- Select IMS Enterprise Suite Version 3.1 and click Continue.
- Select IMS Data Provider for Microsoft .NET and click Continue
  - IMS Data Provider for Microsoft .NET download page
- Select the IMS Data Provider for Microsoft .NET repository file
- Click **Download now** to download the selected files.
- Store the compressed repository file in a accessible location
- You must have IBM Installation Manager Version 1.5.3 or later installed

IMS ES Explorer for Development

03- IMS13 Integration: 186

IMS Enterprise Suite V3.1 Explorer for Development



The IMS™ Enterprise Suite Explorer for Development (IMS Explorer) is an Eclipse-based graphical tool that simplifies IMS application development tasks such as updating IMS database and program definitions, and using standard SQL to manipulate IMS data. Its graphically-driven editors allow the user to display the segment hierarchy for any IMS database, including logical relationships and secondary indexes. It also provides user assistance in the form of rich GUI controls and contextual help to reduce IMS development effort.

The Explorer's graphical editors can be useful for the importing, visualization, and editing of IMS database and program definitions. You can also use the IMS Explorer to easily access and manipulate data stored in IMS by using standard SQL.

IMS Enterprise Suite Explorer for Development

- Enhancements for V3.1 include:
  - Ability to import large numbers of DBDs and PSBs.
  - Automatic imports of referenced DBDs when DBDs and PSBs from the IMS catalog or the host are imported.
  - Ability to import COBOL and PL/I data structures from the host.
  - Support for transaction unit testing.
    - Uses IMS Connect API for Java
    - Can be used in addition to IBM IMS Batch Terminal Simulator
  - Support for IMS catalog navigation.
    - View IMS resources in an IMS catalog-enabled system
    - Import IMS resources into IMS Explorer projects from the view.
    - Show all instances of a given resource or find referenced DBDs or PSBs
  - A Problems View for troubleshooting information
    - Shows resource problems and missing files

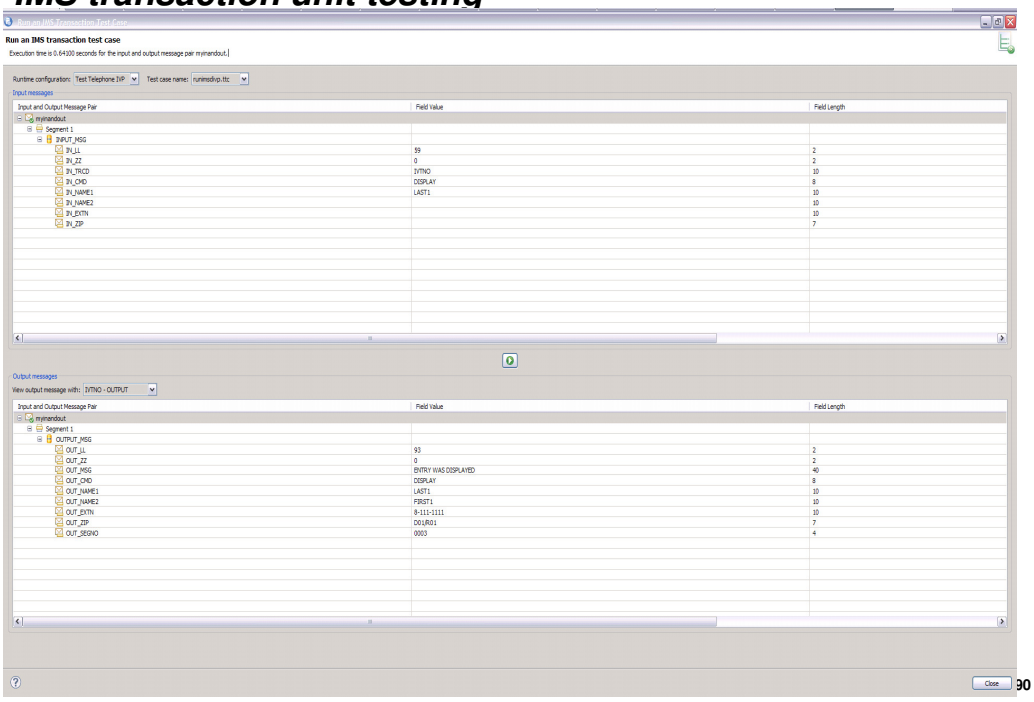
03- IMS13 Integration: 188

Creating and running a unit test case for application debugging

- Create a transaction test project File > New > IMS Explorer Transaction Test Project.
- **Define the input and output messages in the transaction.**
- Create a test case New > IMS Transaction Test Case.
- **Develop a script of the dialog between an existing or future front-end application and one or more transactions running in IMS.**
- Define a IMS server New > Server click IBM > IMS Transaction Server.
- From the test project, right-click and click Run As > Run Configurations.
- **Run the test case.**

You can create unit test cases and provide input message data in human readable format for debugging. After you create a unit test case, you can create variations of it with different input message data, to easily exercise different code paths in the IMS transaction.

IMS transaction unit testing



IMS ES 2.2 Explorer - Importing data structures

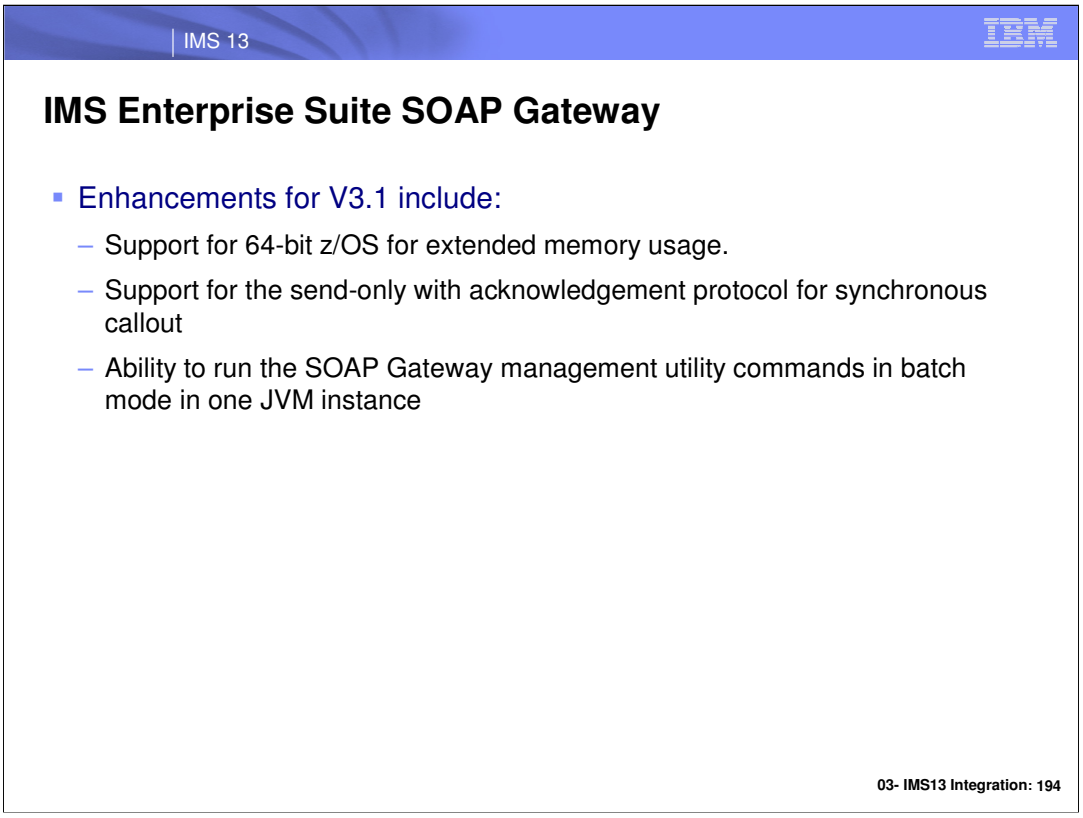
- Importing data structures
  - metadata field layouts imported from data structure files (COBOL copybooks or PL/I includes)
  - Allow COBOL and PL/I importers to directly import data structures from zOS
    - Just like DBD or PSB source import from z/OS
- DBD file
  - select Import COBOL or PL/I Data Structures.
  - select the data structure file that contains the COBOL copybooks or PL/I includes that you want to import.
  - Optional:
    - using source from the IMS catalog, specify the map and case name to import the data structure
- Benefit
  - RDz is not required
  - no need to shell share with RDz

Cross-product integration

- The IMS Explorer supports cross-product integration (shell-sharing) with the following products:
  - IBM® Rational® Developer for System z®
  - IBM Data Studio
  - IBM Problem Determination Tools Plug-ins for Eclipse
  - IBM Explorer for z/OS®
  - IBM CICS Explorer® Software Development Kit (SDK)
  - IBM Rational Team Concert™

03- IMS13 Integration: 192

IMS ES SOAP Gateway



64-bit support for z/OS

SOAP Gateway now runs on the z/OS platform in 64-bit mode, allowing organizations to take advantage of their 64-bit operating environment for extended memory usage.

Send-only with ACK support for synchronous callout

Send-only with acknowledgement protocol support for synchronous callout allows SOAP Gateway to receive a final confirmation that the response message was delivered to the original IMS application that issued the callout request. This confirmation provides SOAP Gateway users additional information about whether a callout response message was sent to IMS and whether IMS received the message.

SOAP Gateway management utility batch mode support

Administrators can now use the batch mode of the management utility to facilitate web service deployment and server management for better performance and manageability. Instead of issuing one command at a time, each with its own JVM instance, you can pass a file with a list of commands to the SOAP Gateway management utility **iogmgmt -batch** command for execution as a batch in one JVM instance.

Enhanced security cipher suite support

SOAP Gateway is enhanced to use the FIPS 140-2 approved cryptographic provider(s); IBMJCEFIPS (certificate 376) and/or IBMJSSEFIPS (certificate 409) for cryptography. The certificates are listed on the NIST web site at http://csrc.nist.gov/cryptval/140-1/1401val2004.htm. SOAP Gateway also adds the support for Transport Layer Security (TLS) V1.2 and for cipher suites with key length of 2048 and key strength of 112 bit, as required by NIST SP800-131a.

Transaction tracking

- SOAP Gateway transaction tracking IDs and logging
  - SOAP Gateway can attach 40 byte *horizontal* tracking ID to inbound web service request
    - ID is sent with the inbound request through IMS Connect to the target IMS application and returned with the response message to SOAP Gateway
  - IMS Connect tracking ID captured by the IMS Connect Event Recorder exit routine (HWSTECL0).
    - This information can be consumed by the IBM IMS Connect Extensions for z/OS and equivalent tools.
    - For IMS 12 requires APAR PM69983 applied to IMS Connect
  - IMS log records for transactions include the tracking ID
    - IBM IMS Performance Analyzer for z/OS and IBM IMS Problem Investigator for z/OS, or equivalent tools, to inspect IMS log records.
  - Benefits
    - Correlates transactions between SOAP Gateway, IMS Connect, and IMS
    - Provides information for diagnostic purposes

SOAP Gateway can now attach a 40-byte message ID to incoming request messages for web services. This ID is sent with the inbound request through IMS Connect to the target IMS application, and is returned with the response message to SOAP Gateway.

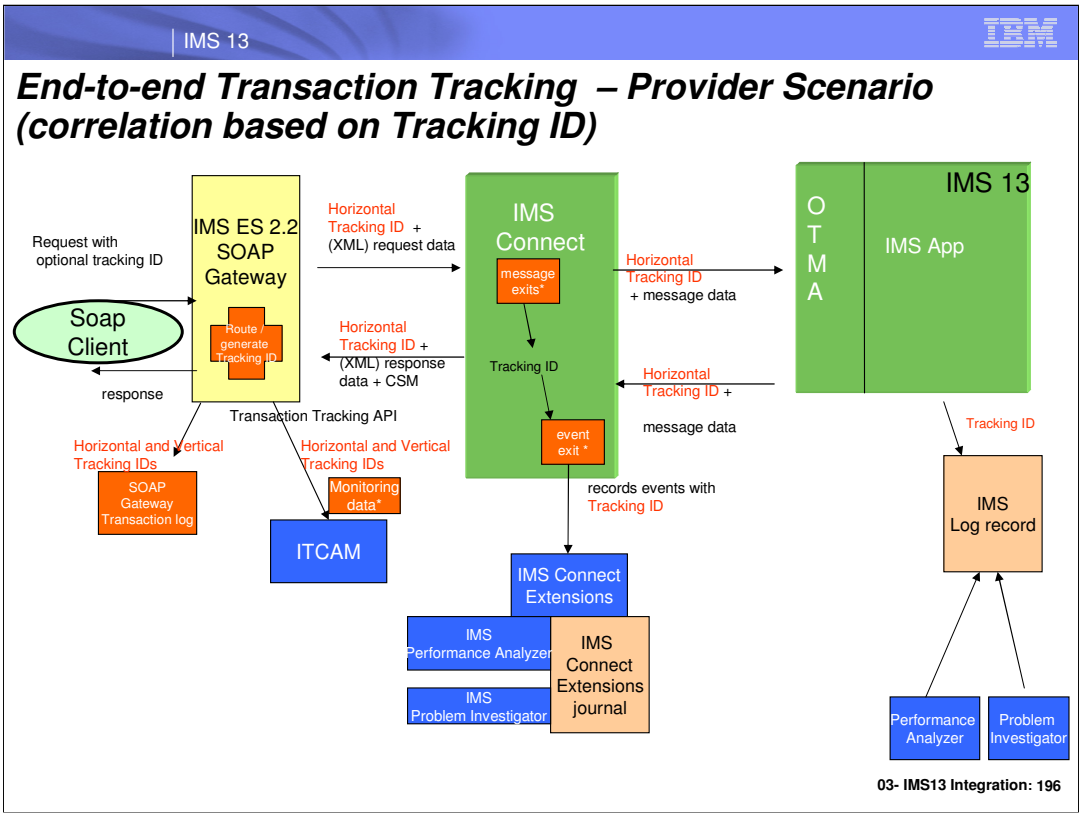
Three different types of message ID are supported:

•SOAP Gateway can get the value of the messageID element in the incoming SOAP message header, and use that value as the message ID.

•SOAP Gateway can get the value of a user-specified element in the incoming SOAP message header, and use that value as the message ID.

•SOAP Gateway can generate a unique ID for every incoming SOAP message.

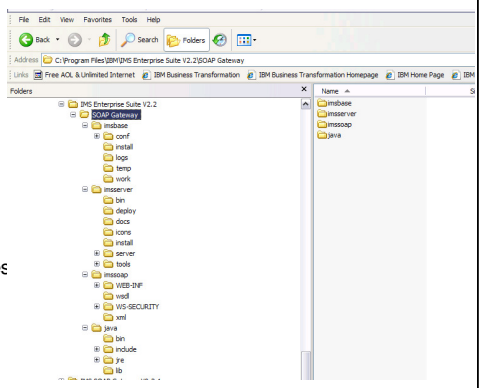
Requirement: IMS 12 with service for APAR PM69983 applied to the target IMS Connect host is required to use horizontal IDs.



IBM Tivoli Composite Application Manager for Transactions (ITCAM) Transaction Tracking API (TTAPI).

IBM Tivoli Composite Application Manager for Transactions (ITCAM) data collector

Advanced installation and maintenance

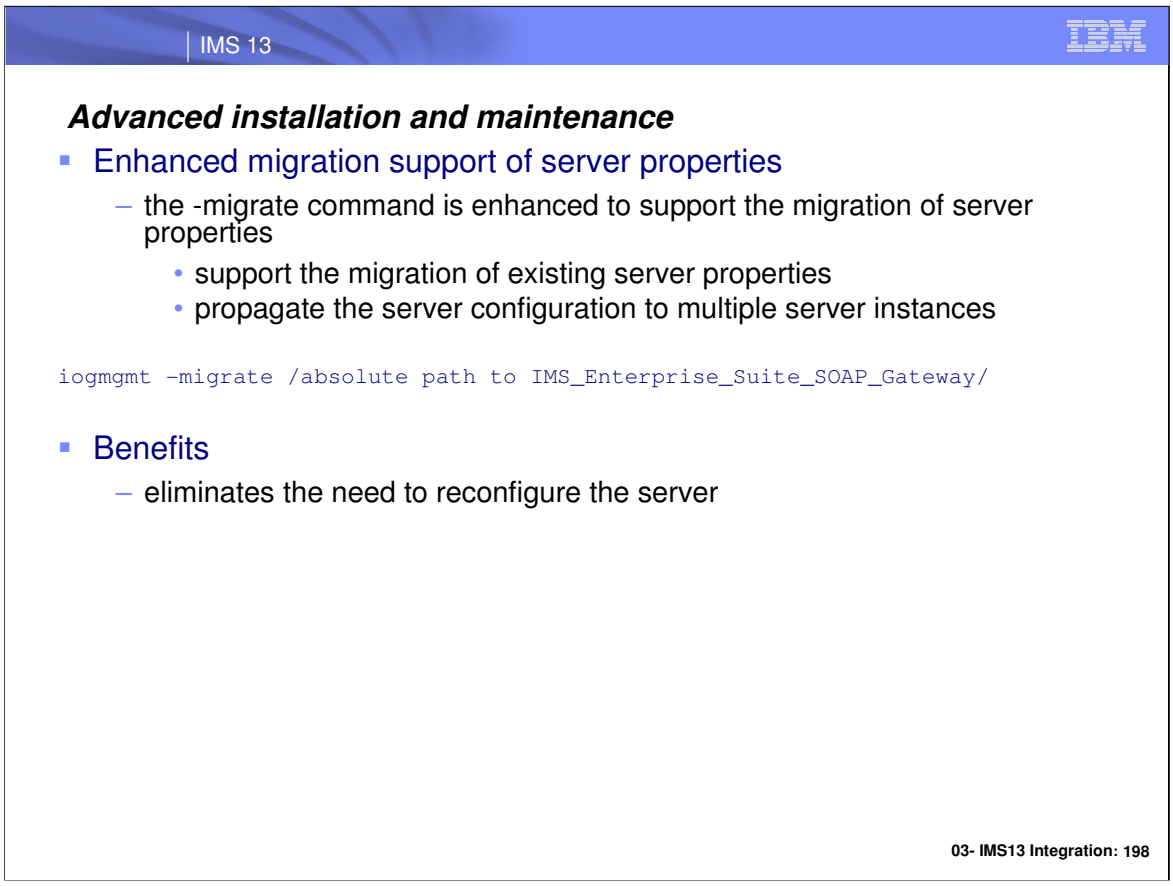


- Server is divided into three components
  - Imsserver – can be mounted as READ only
    - Contains the servers executable code
  - Imsbase – can be mounted as READ and WRITE
    - Contains the servers configurations and log
  - Imssoap - can be mounted as READ and WRITE
    - Contains the user-deployed web service-related files
    - WSDL's , correlators, connection bundles
- Benefits
  - easier to apply maintenance
  - allocate additional disk space when more web services are added
- IBM® Installation Manager for z/OS® V1.5.3
  - Supports centralized repository delivered through the SMP/E process
- Benefits
  - simplifies maintenance process
- Multiple SOAP Gateway server instances
  - share a single instance of the Java™ Virtual Machine (JVM).
- Benefits
  - reduces the amount of storage required for each additional server instance

An installation of IMS Enterprise Suite Version 2.2 SOAP Gateway consists of three parts that can be installed in different directories (or mount points on z/OS). This three-part architecture separates the binary files that run the SOAP Gateway server and the management utility from server configuration files and user files such as

web services-related artifact files. This separation makes it easier to apply maintenance and allocate additional disk space when more web services are added.

For z/OS the SOAP Gateway installation requires IBM Installation Manager V1.5.3. Installation manager simplifies maintenance by allowing the installing and upgrading of the server by pulling from a centralized repository that is delivered through the SMP/E process.



The iogmgmt -migrate now supports the migration of server properties. To migrate from version 2.1, specify the absolute path to the installation of IMS Enterprise Suite Version 2.1.

Clone creates a copy of the web services and server properties from a master Version 2.2 server.

The correlator schema has changed in IMS Enterprise Suite Version 2.2 SOAP Gateway. When you upgrade to IMS Enterprise Suite Version 2.2, the process of migrating existing web services iogmgmt -migrate handles the correlator migration.

If the installation directory for the **imsserver** component is read-only, you must change it to read/write mode before you run the migration tool.

WS-Security

- WS-Security SAML unsigned tokens for synchronous callout applications
  - Originating Userid (PSTUSID) for the IMS synchronous callout application is passed to the external web service for further authentication and authorization
- Benefit
  - Provides message-level security for synchronous callout
- WS-Security enhancement for provider web services
  - support for Security Assertion Markup Language (SAML) 2.0 sender-vouches signed tokens
- Benefit
  - Provides additional message integrity for service provider processing
  - Extends SOAP Gateway support of WS-Security standards

For the synchronous callout scenarios, in addition to transport-level security through basic authentication, server authentication, or mutual authentication, SOAP Gateway now supports message-level security with SAML 1.1 and SAML 2.0 sender-vouches unsigned tokens.

SAML is an XML-based standard developed by Security Services Technical Committee (SSTC) of Organization for the Advancement of Structured Information Standards (OASIS). This standard facilitates:

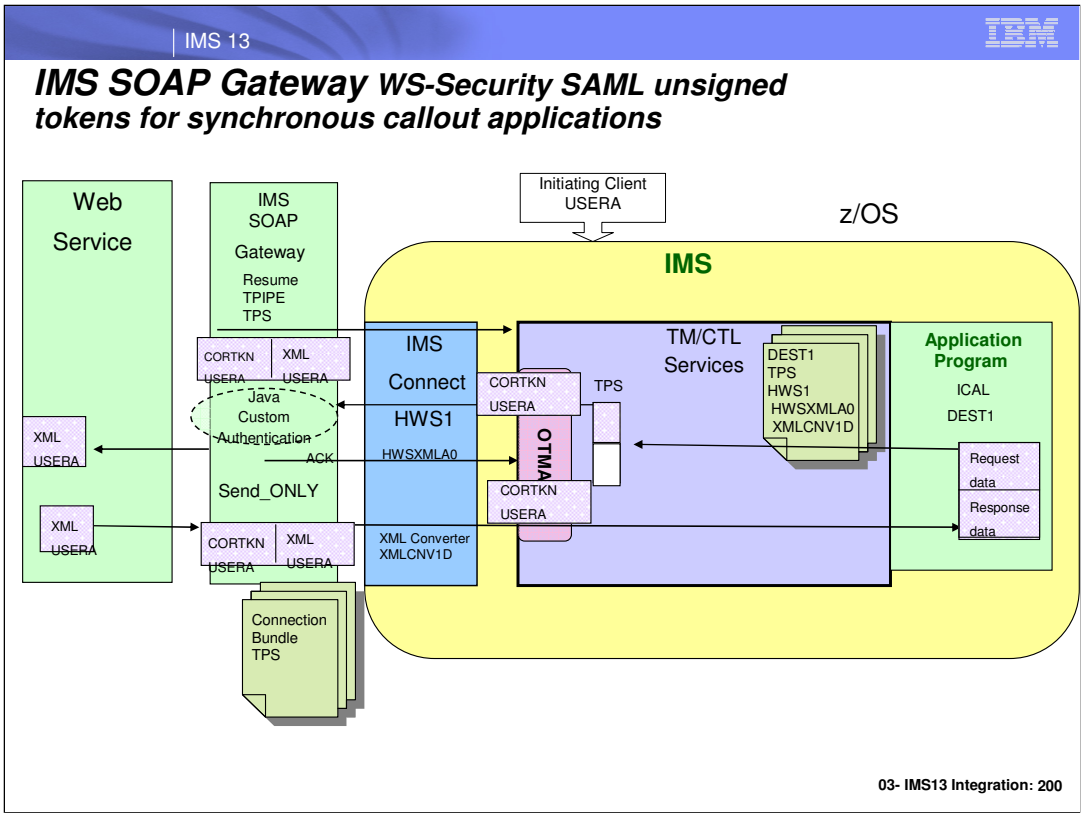
•The exchange of user identity and security attributes information between communicating parties at the SOAP message level.

•The exchange of authentication and authorization assertions across web service transactions.

WS-Security SAML confirmation method is supported for synchronous callout applications by extracting the user ID (the user that initiates the synchronous callout application)

from the correlation token and passing it to the external web service.

SOAP Gateway also supports custom authentication modules for accessing the security header for validation before the SOAP request messages are sent out to the external web service server.



SOAP Gateway message-level security with Security Assertion Markup Language (SAML) 1.1 and SAML 2.0 sender-vouches unsigned tokens.

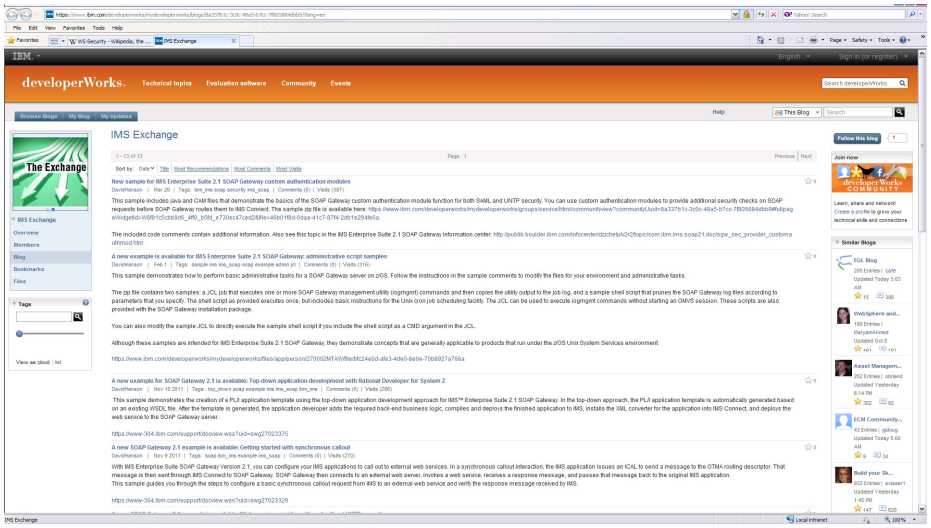
The ID of the user who initially invokes the IMS synchronous callout application is obtained from PSTUSID and moved into the synchronous callout correlator token field (`COR_USERID`) as the web service client which is passed in the SOAP header to the external web service for further authentication and authorization.

SOAP Gateway also supports custom authentication modules for accessing the security header for validation before the SOAP request messages are sent to the external web service server.

The IMS application issues the ICAL call to send the callout request data ,the OTMA descriptor name and optional timeout value. A correlation token including the initiating client userid will be sent together with the callout request which is managed by IMS SOAP Gateway. IMS SOAP Gateway looks up the callout correlator and the WSDL file based on the Web service correlation information in the callout request message. The outbound SOAP request will be built based on the correlation and WSDL file information to invoke the external web service provider. This includes obtaining the initiating client User ID from the correlation token and setting it in the SOAP Envelope security header for the XML document

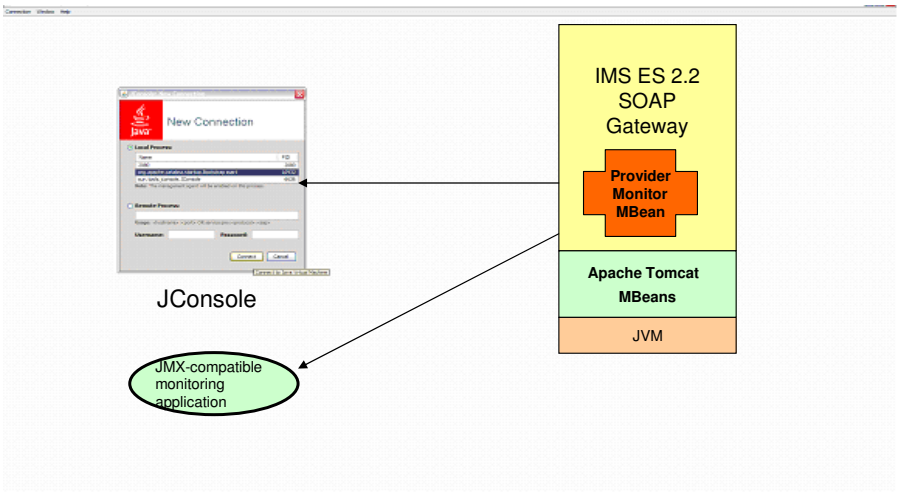
IMS ES 2.2 SOAP Gateway New Samples

- IMS Exchange web site updated
- Link on IMS Enterprise Suite SOAP Gateway web page



JMX MBean interface for web service provider monitoring

- C:..\IMS Enterprise Suite V2.2\SOAP Gateway\java\bin
  - JConsole.exe



JMX MBean interface for web service provider monitoring

In addition to the standard JMX instrumentation for the SOAP Gateway JVM, a customized MBean interface for SOAP Gateway SOAPGatewayProviderMonitorMBean provides statistics about SOAP Gateway web services activity, connection bundles, and connections to IMS Connect.

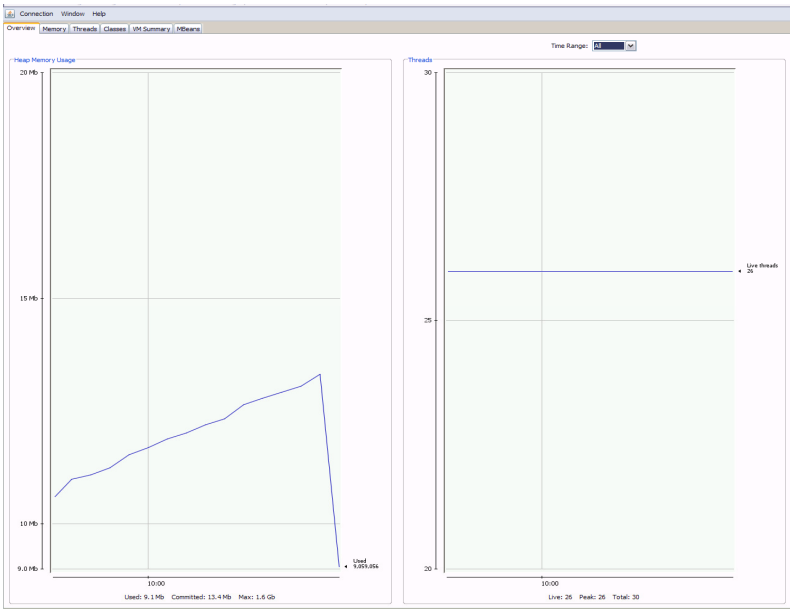
JMX-compatible monitoring application

Apache Tomcat 7.0 MBeans - The Apache Tomcat servlet container, Catalina, is instrumented with JMX MBeans.

org.apache.catalina.startup.Bootstrap - basic JVM information including heap memory usage, thread count, loaded classes, and CPU utilization.

MBeans tab.- expand the folder **com.ibm.ims.soap.server**, then the node **SOAPGatewayProviderMonitorMBean**, and then click **Operations**.

JConsole sample output



03- IMS13 Integration: 203

IMS ES Connect API for Java

IMS ES Connect API for Java

- IMS and IMS Connect type-2 commands
- Performance Enhancements
- Support for SendOnly synchronous callout response messages with acknowledgement
  - Function requires that both the following APAR/PTFs are applied
    - IMS Connect 12: PM39569/UK74666
    - IMS OTMA 12: PM39562/UK74653
- Benefits
  - Custom written IMS Connect TCP/IP Java client applications
    - Can send and receive commands to IMS and IMS Connect
    - Can request and receive an indication of response delivery to IMS for synchronous callout processing

Function requires that both the following APAR/PTFs are applied:
• IMS Connect 12: PM39569/UK74666
• IMS OTMA 12: PM39562/UK74653

New AIB field - AIBUTKN

Provides optional specification of a 1-8 byte map name included in the OTMA state data prefix to be sent to the callout destination.

IMS 12: PM73135

Sample application program for type 2 commands

samples.com.MyCompany.MyProjects.Type2Command.Type2CommandSample.java

```
myTmInteraction.setImsDatastoreName("MYDATASTORENAME"); myTmInteraction.setInteractionTypeDescription
  (ApiProperties.INTERACTION_TYPE_DESC_TYPE2_COMMAND);
          myTmInteraction.setAckNakProvider(ApiProperties.API_INTERNAL_ACK);
          myTmInteraction.setImsConnectCodepage("cp037"); // EBCDIC
          myImsConnectCodepage = myTmInteraction.getImsConnectCodepage();
       // get InputMessage instance from myTMInteraction
          inMsg = myTmInteraction.getInputMessage();
          inData2DByteArray = new byte[1][];
          inData2DByteArray[0] =
          new String("CMD(QUERY TRAN NAME(IVTNO) SHOW (ALL))").
            getBytes(myImsConnectCodepage);
      // Populate InputMessage object with input byte array
          inMsg.setInputMessageData(inData2DByteArray);
      // Execute interaction
          myTmInteraction.execute();
     // Get output from myTMInteraction
          outMsg = myTmInteraction.getOutputMessage();
         if((outMsg.getImsConnectReturnCode() == ApiProperties.IMS_CONNECT_RETURN_CODE_SUCCESS) &&
             (outMsg.getImsConnectReasonCode() == ApiProperties.IMS_CONNECT_REASON_CODE_SUCCESS))
      {
          // Get the type-2 command response object
              Type2CmdResponse myT2CmdRsp = outMsg.getType2CommandResponse();
```

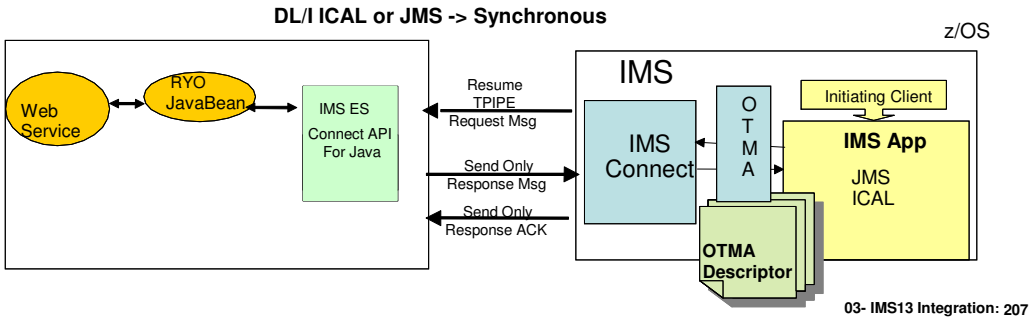

Sample code

IMS ES Connect API for Java

- SendOnly synchronous callout response messages with acknowledgement
  - client application gets acknowledgement when the response message is received by IMS

```
INTERACTION_TYPE_DESC_SENDONLYACK_CALLOUT_RESPONSE
```

IMS Service Consumer



The send-only protocol for synchronous callout responses can include an acknowledgement to the callout response so the client does not need to switch to receive state after sending the response to IMS Connect

The IMS Enterprise Suite 2.2 Connect API for Java™ includes support for send-only synchronous callout response messages with acknowledgement.

The new interaction type description is set with the following method call for a TmInteraction object:myTmInteraction.setInteractionTypeDescription (INTERACTION_TYPE_DESC_SENDONLYACK_CALLOUT_RESPONSE)

This feature allows a client application to get an explicit acknowledgement when the response message is received by IMS, compared to a normal send-only response message that does not get any receipt confirmation.

The client receives the acknowledgement after calling the execute() method of the TmInteraction object. After the acknowledgement is received, the return code and reason code from the request status message (RSM) can be retrieved with the existing getImsConnectReturnCode() and getImsConnectReasonCode() methods. If the acknowledgement indicates a successful message in complete status message (CSM) format, both values are 0.

This interaction type is comparable to the existing INTERACTION_TYPE_DESC_SENDONLYACK interaction type, but it is applicable to synchronous callout response messages instead of requests that are initiated from the Connect API for Java client application.

Performance data

- For inbound, 16,000 transactions per second using V2R2 which is a 3% improvement over V2R1
- For callout, 11,000 transactions per second using V2R2 which is a 56% improvement over V2R1

A base line was established for Inbound and Callout with IMS Enterprise Suite V2R2's Connect API for Java. We were able to reach 16,000 transactions per second for inbound using V2R2 IMS Connect API for Java code. The average CPU % used for LPAR1 with 5 CP's running IMS and IMS connect was 58% and for LPAR2 with 6 CP's running the IMS Connect API for Java client code was 13.52%. 100 clients
Using the same environment for outbound, we saw 11,000 ICAL requests/sec with CPU% used for LPAR1 at 78.57% and LPAR 2 at 12.12%. 10 TPIPE's and 10 response threads were used for Callout. 150 TPNS clients drove the COBOL IMS echo application sending messages on the TPIPEs.

Based on our performance measurements, ES 2.2 IMS Connect API for Java impacts the total cost of ownership by reducing the CPU usage.

The improvement of going from V2R1 to V2R2 was 3% for inbound and 56% for outbound.

IBM IMS Explorer for Administration

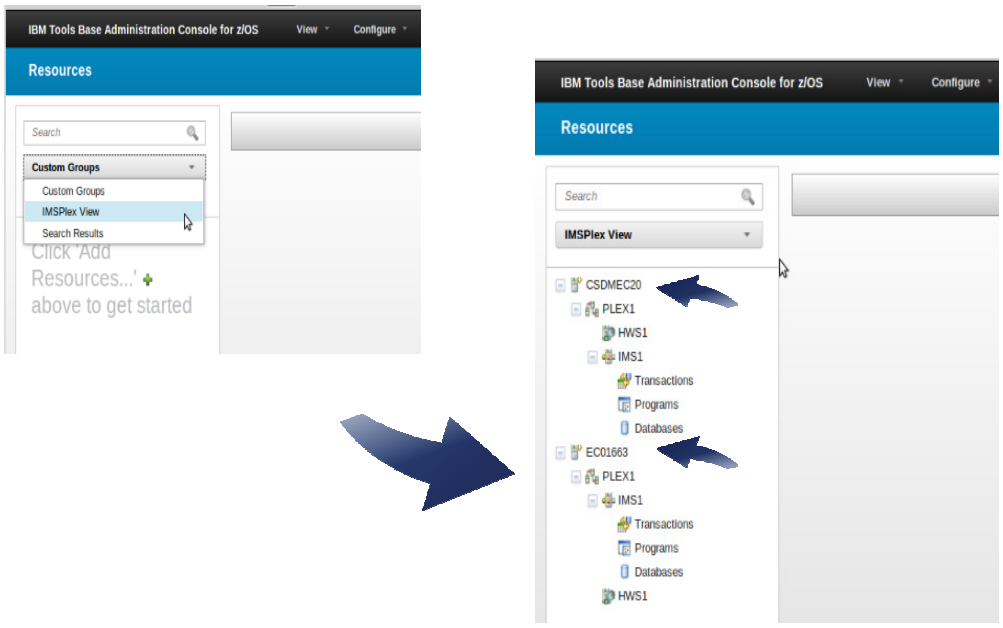
03- IMS13 Integration: 209

IBM IMS Explorer for Administration

- Web-based console to operate and administer IMS
  - Graphically administer IMS Databases and Transactions
    - Connects to the IMS Operations Manager through IMS Connect
  - Discover IMS resources
  - Show the health of the resources
  - query, start, and stop IMS resources
  - View IMSPlex
    - visualize relationships between various IMS resources
      - transactions, programs and databases in one view
  - Replacement for IMS Control Center
- Software requirements
  - IMS Explorer for Administration is available through
    - APAR PM94292 extension of the Administration Console component
      - IBM Tools Base for z/OS , V1.4 (5655-V93)
  - IMS Version 12
  - IMS Connect
  - Common Service Layer OM and SCI
  - Supported web browser
    - Firefox, Internet Explorer, Safari

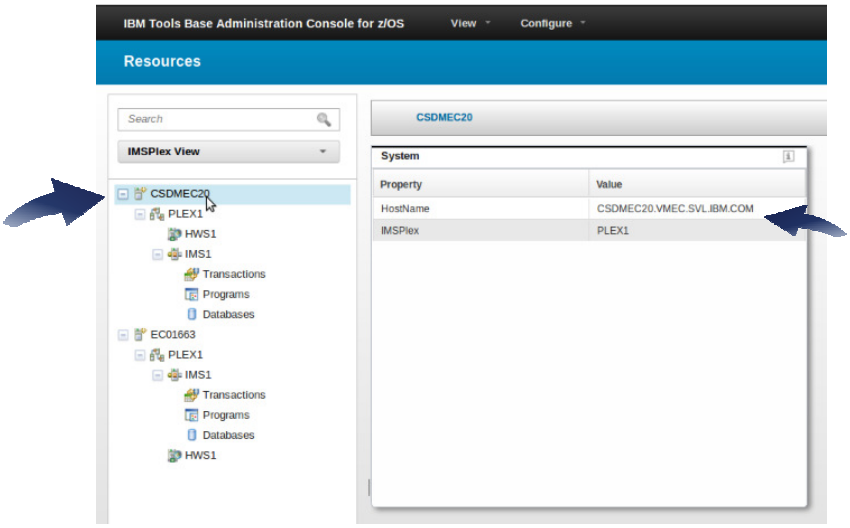
The Explorer for Administration provides accessibility from any supported web browser to manage IMS resources

Explorer for Administration – Select the Resource



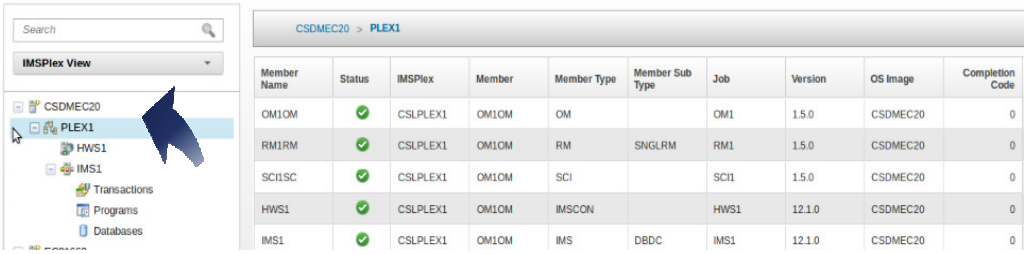
Select from custom groups the IMSPlex view and notice on the right you will have two system nodes; here we have configured two LPARs to the two therefore see two system nodes.

Explorer for Administration – System View(s)



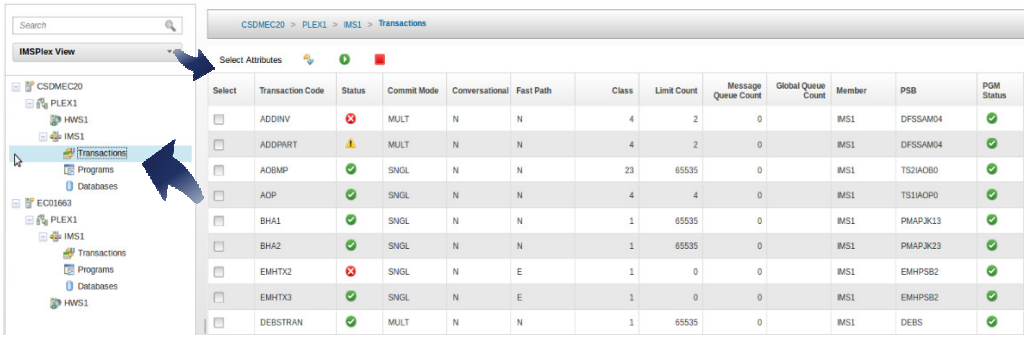
Clicking on the system nodes shows you the hostname and the IMSPlex for the systems. If you have more than one IMSPlex they will appear under the system node.

Explorer for Administration – IMSPlex Resources



In the navigation tree on the left you can see the IMSPlex “PLEX1”, when you click on it the resources part of the IMSPlex appear on the right. Notice the status indicated as green and the various members of the IMSPlex (SCI, OM, ICON, IMS, RM) and their attributes such as resource versions.

Explorer for Administration – Manage Transactions



In the navigation tree if you expand it till you see your IMS, you can drill down further into the IMS's transactions, programs and databases. Here we are highlight in the tree the transactions.

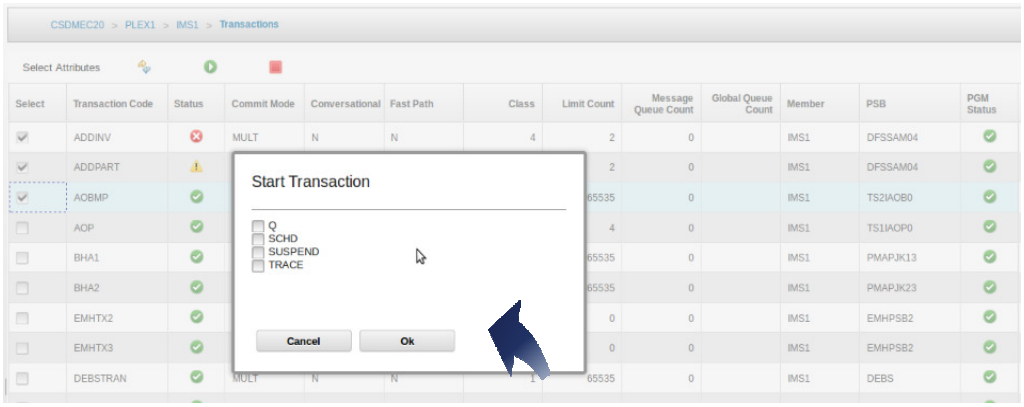
Notice in the transactions main page you also see the related programs on the right. Hovering over the red, yellow or green icon will show you the status's of the transaction.

The default columns are predefined attributes but can be altered by clicking on the attributes button above the transaction rows and submitting your change. You can select any IMS Attributes that return from the QUERY Tran TYPE2 command.

Also notice you can multi select a group of transactions and start or stop them.

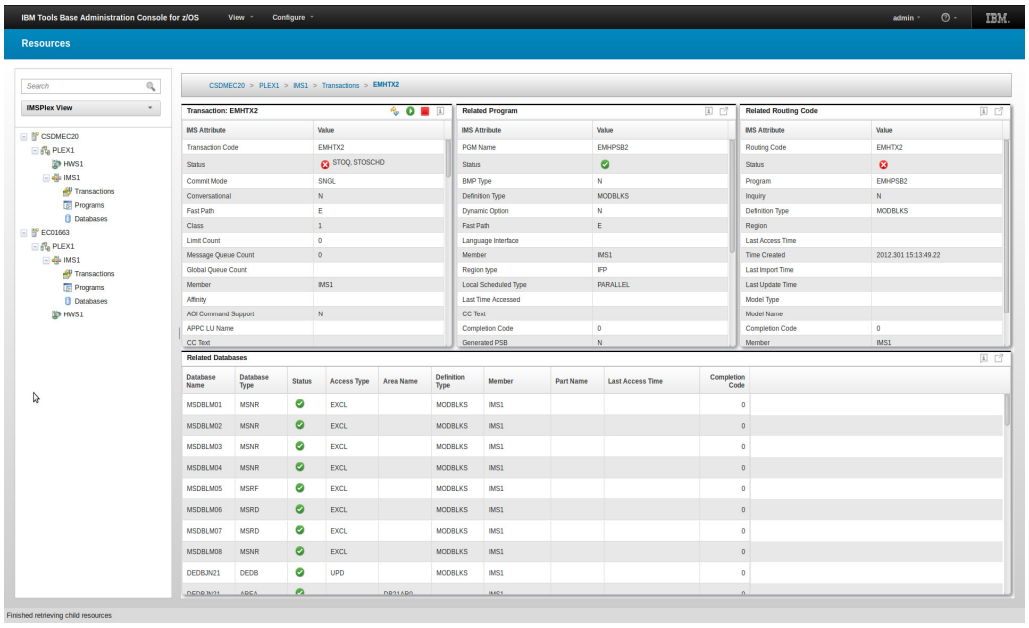
If you double click on any transaction row, you will be taken to the transaction details and relationship page where you will see more details about this transaction and its relationships.

Explorer for Administration – Start/Stop Transactions



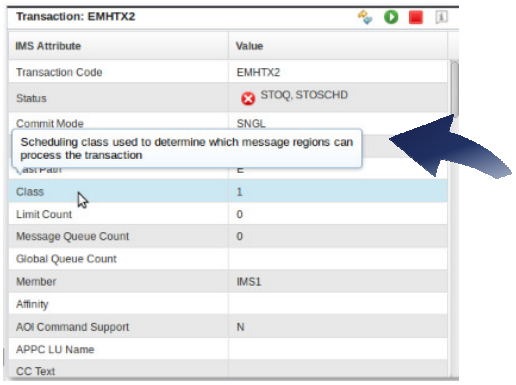
Start/Stop multiple transactions.

Explorer for Administration – Transaction Details



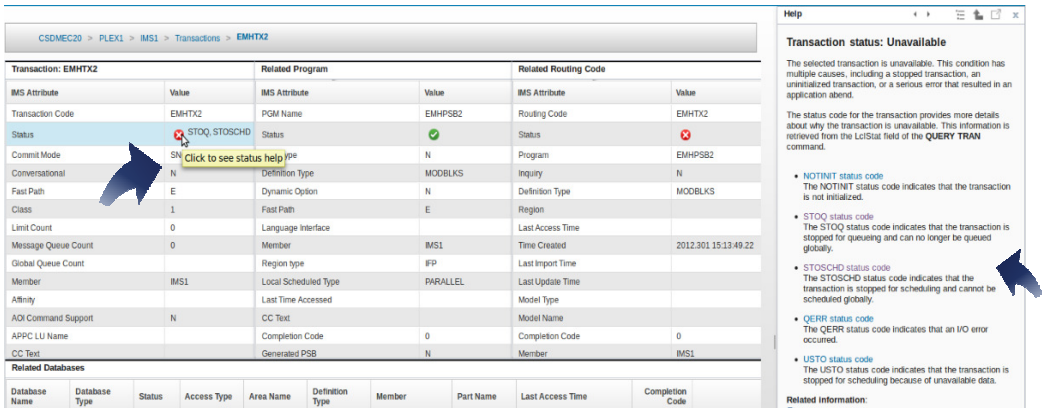
Transaction details view. Here you can trill into the transaction from the main page by double clicking the transaction row EMHTX2 and you see the transactions details and its related programs, routing codes and databases.
Notice that there is context sensitive help on this page with regard to each of the panels, see later slides for examples.

Explorer for Administration – Hover Help



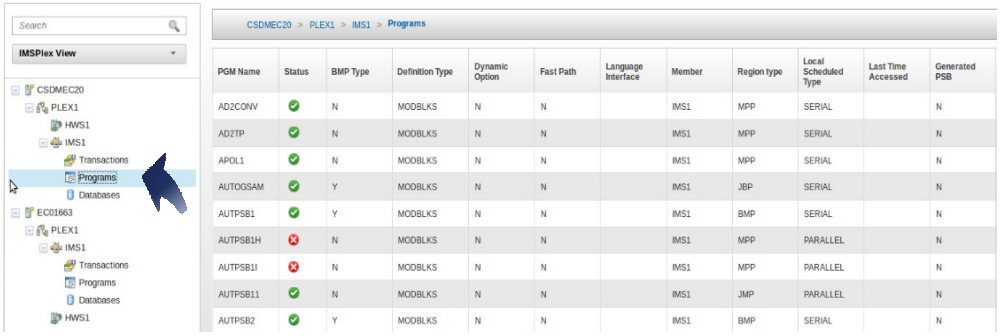
Hover help is available to help assist you with unfamiliar attributes. Clicking the “i” in the top right transaction panel would also display a help panel with more details for transaction attribute explanations. Notice that the hover help and the “i” help will only be related to the panel you are in to reduce the need to search numerous pages from the IMS book.

Explorer for Administration – Context Sensitive Help



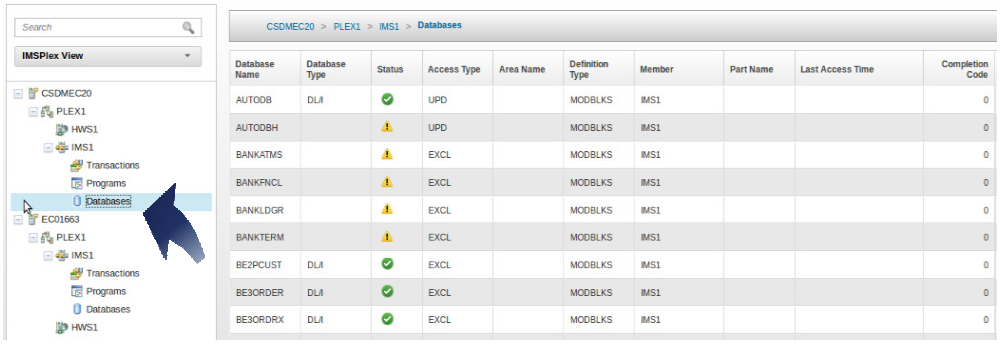
Notice next to the red icon the status codes shown indicating why its a red color. If you want to know more about these particular status's or why its marked red click on the icon and on the right side of the browser a help panel appears with related help to this transactions status. Click on one of the transaction status links in the help panel and drill further to see corrective actions.

Explorer for Administration – Programs View



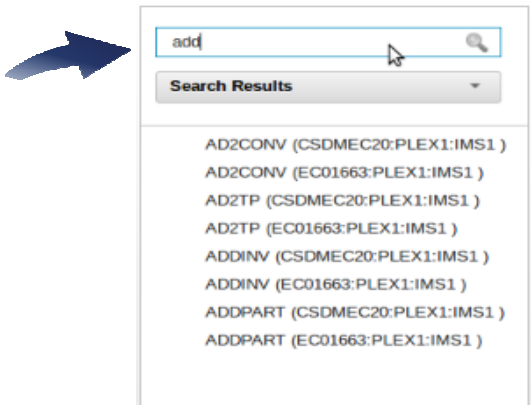
Similar to transactions there is the ability to display information and status's about programs with the double click function that will take you into a details and relationship view.

Explorer for Administration – Databases View



Similar to transactions there is the ability to display information and status's about databases with the double click function that will take you into a details and relationship view.

Explorer for Administration – Search



If you know the name of the transaction, program or database you can search for it in the top left corner of the browser where there is a text field. Even if you don't know exactly the name, you can search character by character and the search will narrow down eventually to the name of the resource you are searching for. Clicking on it will take you to the respective details view for that resource. IE if it's a transaction you searched on, you will see the transaction details for that transaction and related programs and databases.

System Prerequisites

- Software requirements
  - IMS Tools Admin Console (no-charge)
  - IMS Version 12 or later
    - IMS Connect
    - Common Service Layer OM and SCI
  - Supported web browser
    - Firefox, Internet Explorer, Safari