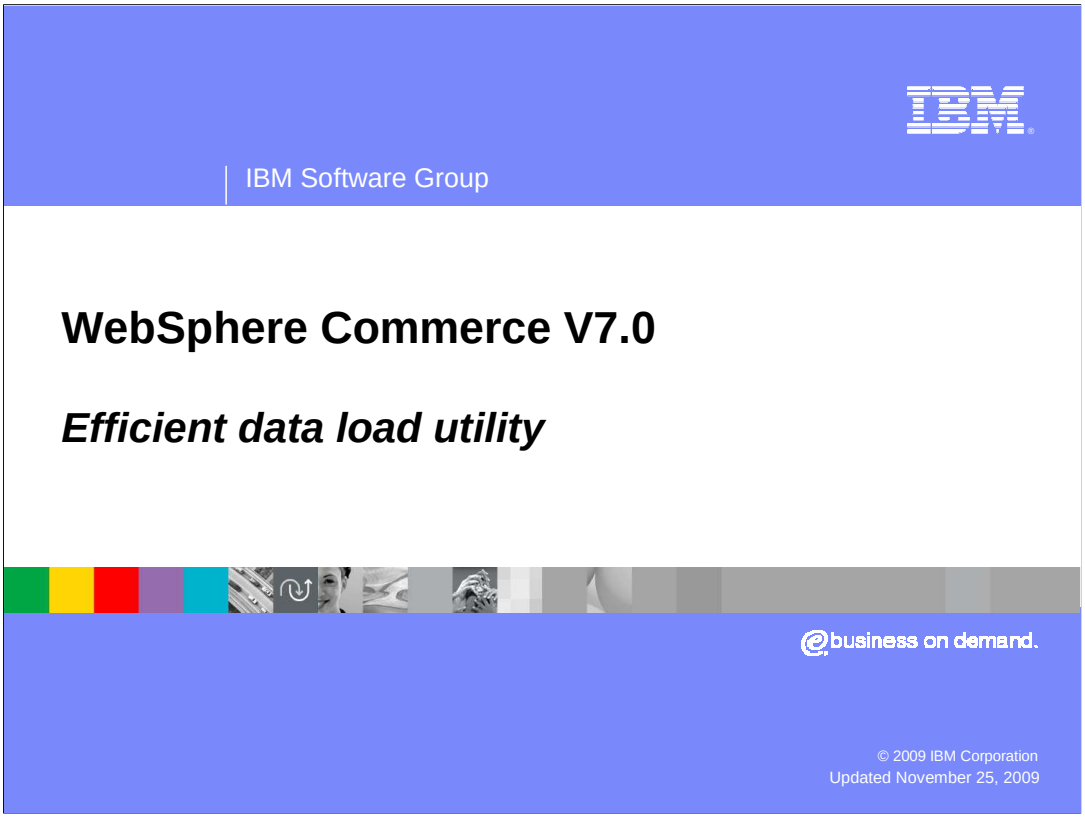


This presentation discusses the efficient data load utility.

Goals

- Understand new data load framework
- Understand how to run data load utility

This presentation will help you understand the new data load framework and how to run the data load utility.

Agenda

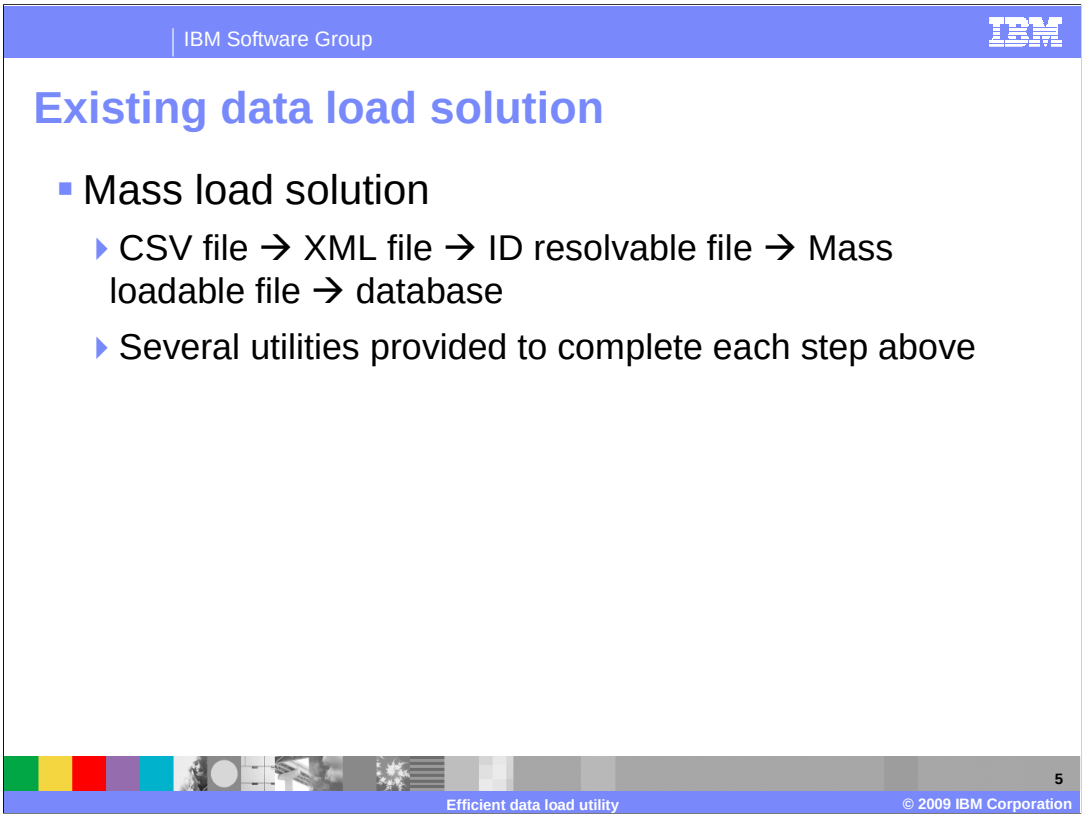
- Data load solution overview
- Data load framework
- Data load configuration files
- Data load command and options
- Custom extension tables

This presentation discusses the data load solution overview and the data load framework. The configuration files and command line options show you how to run the data load utility. Finally, loading custom extension tables is discussed.

Definition of data load

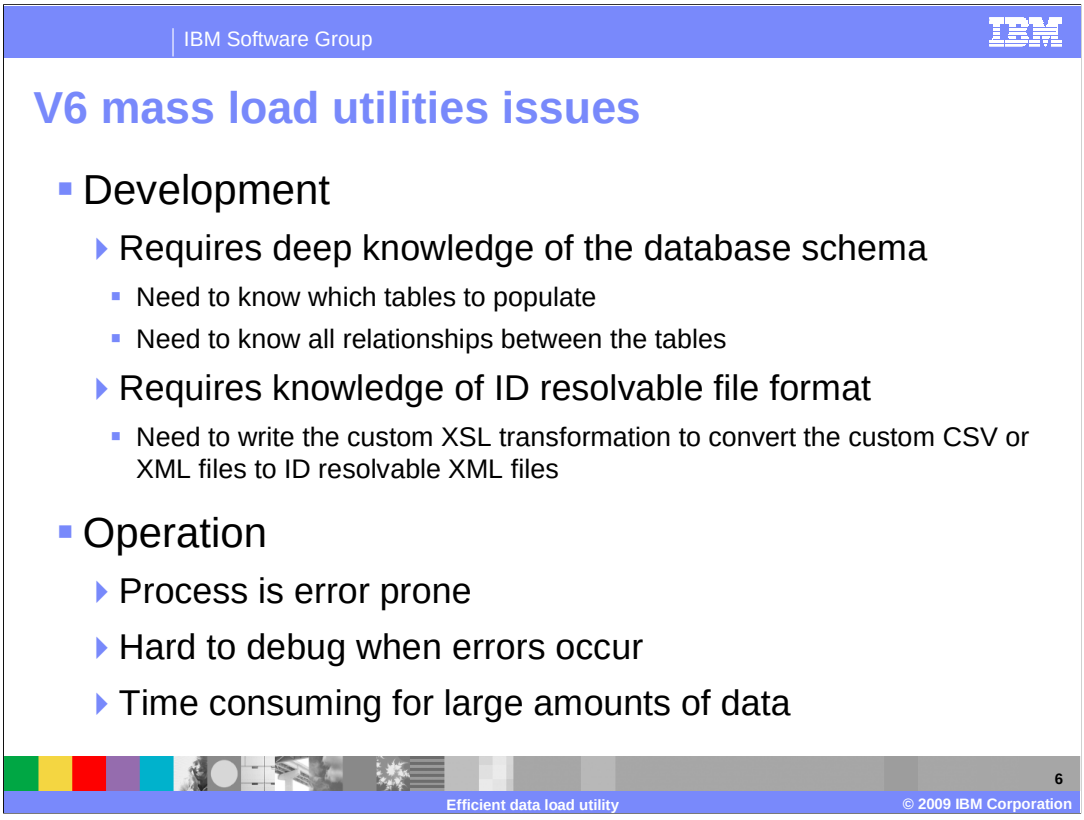
- Initial load
  - Occurs once
  - Large amount of data
- Delta load
  - Occurs regularly
  - Adjustments to initial load
  - Small amount of data

Data load means to load the data from external data sources to the WebSphere® Commerce database. There are typically two scenarios for data loading. Initial data loading is the first time you load the data into the database. Typically a large amount of data is involved. Delta load is used for data insert, update and delete. Delta load can happen daily or weekly. There is a smaller amount of data involved compared to the initial load.



For many years, mass load has been used to load data in WebSphere Commerce. The mass load solution accomplishes loading in several steps. If your input data is a CSV file, you need to convert it to an XML file. The XML file then needs to be transformed to a WebSphere Commerce ID resolvable XML file format by using XSLT (**Extensible Stylesheet Language Transformations**). Next you run the **ID resolver** to generate the database keys. Finally, you run mass load to load the data into the WebSphere Commerce database.

Each of these steps uses a utility to do the required transformations.

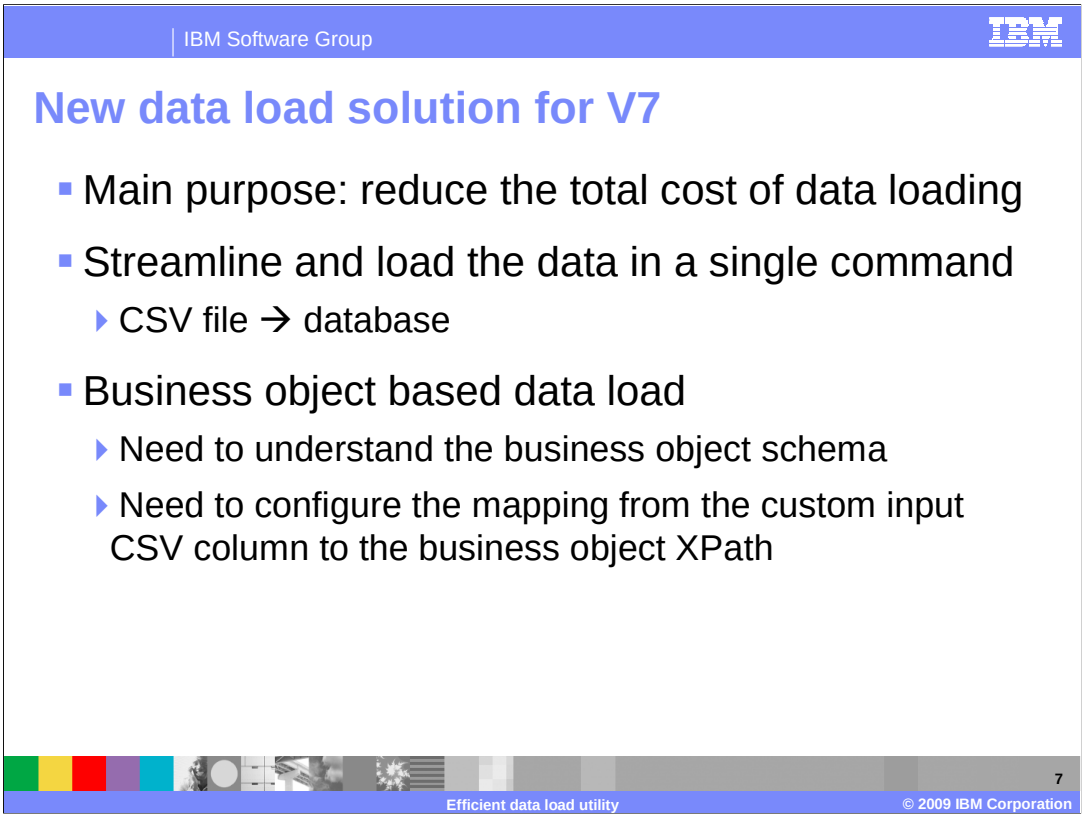


There are several issues with the V6 mass load utilities.

In order to successfully use the utilities, you require a deep understanding of the database schema. You need to know which tables your data resides in and the relationships among these tables. You also need to know the hidden tables which are used by the WebSphere Commerce to do internal bookkeeping. These hidden tables are required to be included into your input XML file.

You also need to know the ID resolvable file format. You need to write XSL transformation code to convert your XML file to ID resolvable XML format files.

The current data load process requires running multiple utilities in sequence to load the data. Because of the number of processes involved, it is hard to isolate which steps caused a problem when an error occurs. There are performance implications due to the intermediate files which are written and then read again.



A new data load solution is provided in WebSphere Commerce V7. The main purpose of this solution is to reduce the total cost of the loading data.

This new data load solution streamlines the process. A single command reads the input data and writes it to the database. There is no need to generate intermediate files.

This new data load solution is based on the WebSphere Commerce business object model. To use this new data load utility, you need to understand the WebSphere Commerce business object schema instead of the physical database schema. You only need to have CSV input files, and then map the columns in the CSV file to the components of the logical business objects using XPath notation.

Benefits of new data load utility

- High performance
- Scalability
- Customizable and extendable
- Transaction commit or rollback within the business object boundary
- Better diagnostics and error reporting

The performance is much better because of the elimination of reading and writing intermediate files.

The new utility is also more scalable. You can load very large input data either for the initial load or the delta load.

The new data load utility is customizable and extendable. It allows you to provide your own implementations in each layer of the data load execution flow.

Because this new data load is based on business objects, transaction commit and rollback is within the business object boundary instead of database tables. This type of transaction boundary ensures that the data is consistent. Either all the business object related tables are loaded, or, if a failure occurs, none of the tables are loaded.

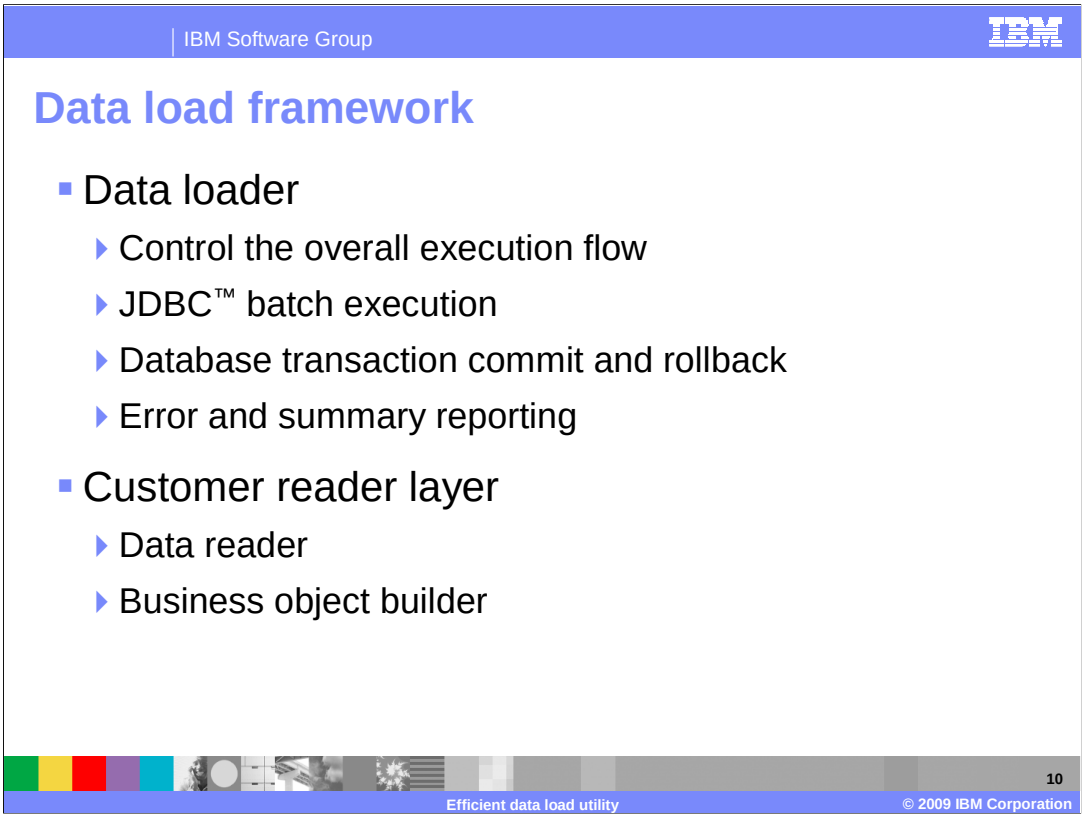
In the event of a failure, more specific error reporting improves the problem determination process. The error reporting not only tells you whether the input CSV files or configuration files contain the error, but also tells the specific place where the error is.

Business object limitations

- Business objects supported
  - Catalog
  - Price
  - Inventory
- Use mass load utilities for other objects

In WebSphere Commerce V7, data load supports catalog and catalog related components, such as catalog groups, catalog entries, merchandise associations, price, and inventory for the catalog entries.

In WebSphere Commerce version 7, you can still use the mass load facility. If you want to load components other than catalog, price and inventory, you can either customize the data load utility or keep using the mass load utilities.



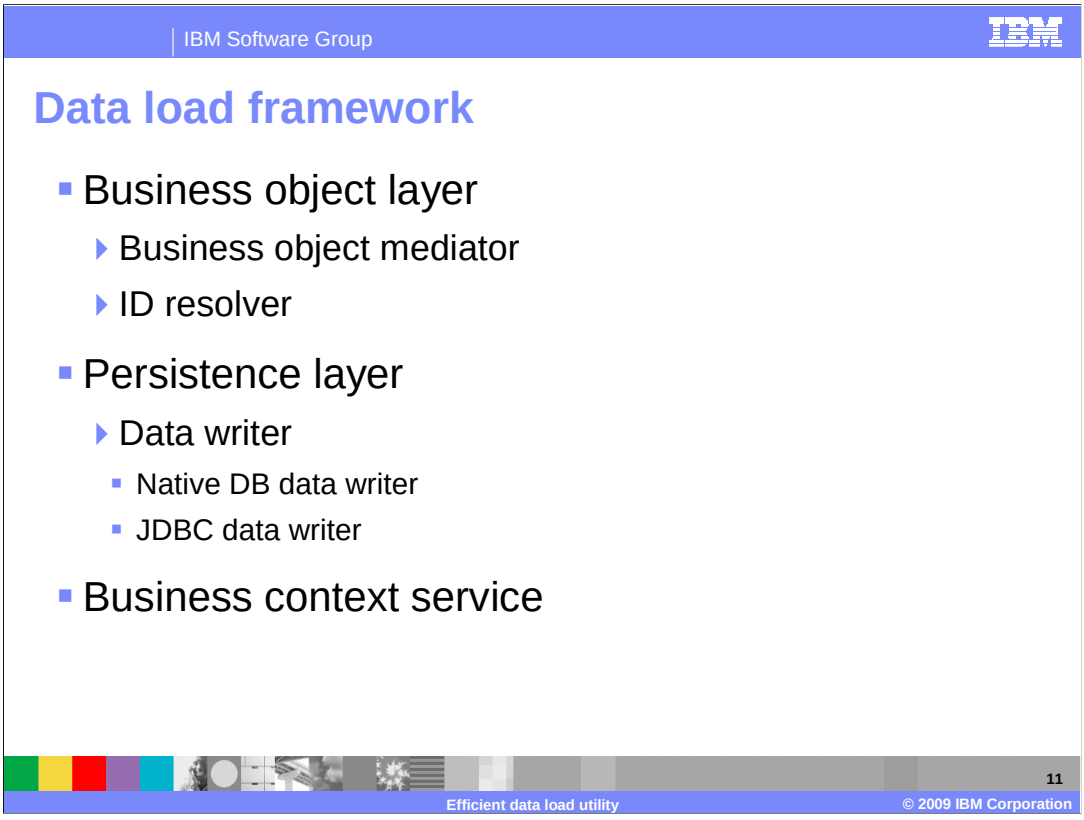
The new data load framework contains several components.

The **Data Loader** component controls the overall data load execution. It controls the database transaction and rollback. It decides when the JDBC batch API should be called. It is also responsible for generating the error message and summary report.

The customer reader layer contains two components: the data reader and the business object builder.

The **data reader** is responsible for reading in the original physical data and passing it to the business object builder.

The **business object builder** is responsible for converting the data from the data reader into the WebSphere Commerce business objects.



The **business object layer** is responsible for converting the business objects into physical objects which later can be persisted into the database.

Within the business object layer, the **business object mediator** is responsible for converting the business object into a list of physical objects, which are also called service data objects.

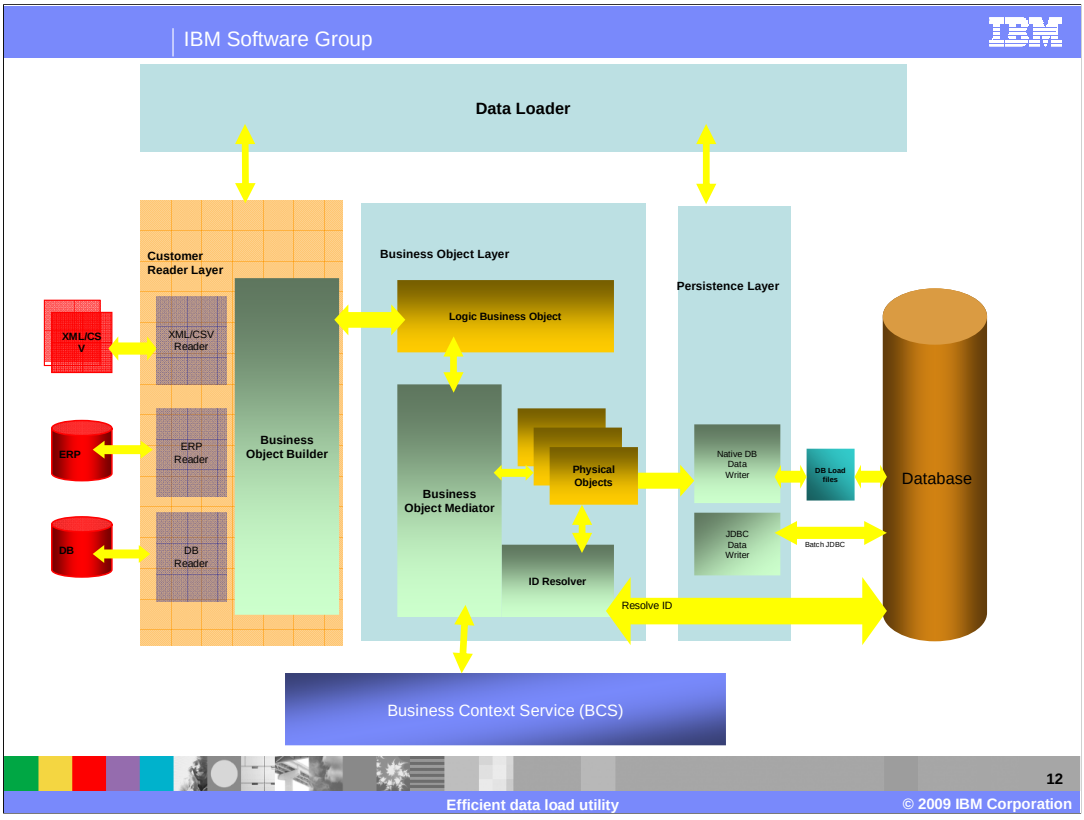
The **ID resolver** is responsible for resolving the generated primary key which is used by the business object mediator to populate the primary key in the physical objects.

The **persistence layer** is responsible for persisting the physical objects into the database. This is done by the data writer component. There are two types of data writers provided.

The **native DB data writer** is used to persist the physical objects into a list of files in a database native loadable format. These files can be loaded into the database by the database native load utility. The **native DB data writer** is primarily used in the initial data load with large input data. The native load provides better performance for large input data, compared to the JDBC data writer.

The **JDBC data writer** is used to persist the physical objects into the database directly using the JDBC batch APIs.

The **business context service** is used in the above components when the business context data is needed.



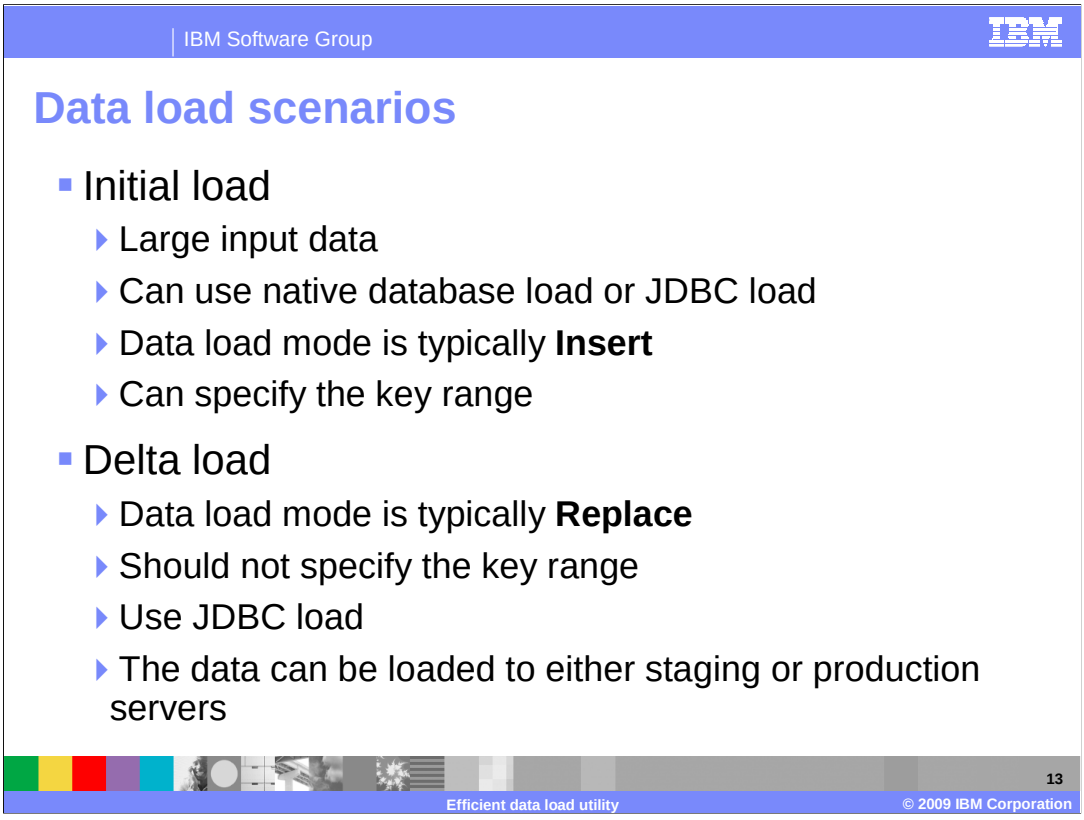
The diagram in this slide shows all the data load components and the high level data load flow.

The data loader calls the data reader to read in a record. If the data reader is a CSV reader, it will read in one line from a CSV file. For version 7, only the CSV reader is provided.

The data loader passes the record to the business object builder. The business object builder is responsible for populating a logical business object based on the input record and a loader configuration file. This configuration file defines how to map the business object components to the input file data using an XPath mapping.

The business object builder passes the logical business object to the business object mediator. The business object mediator is responsible for converting the logical business object to a set of physical objects which are mapped to the physical database schema. Each business object has a corresponding business object mediator to do the conversion. You can write your own business object mediator.

The business object mediator passes the list of physical objects to a data writer to do the persistence. The data writer can be a JDBC data writer or a native DB data writer. The JDBC data writer will load the physical objects to the database directly. The Native DB Data Writer will save the physical objects to native database loadable files. These files can be loaded by the database native load utility.

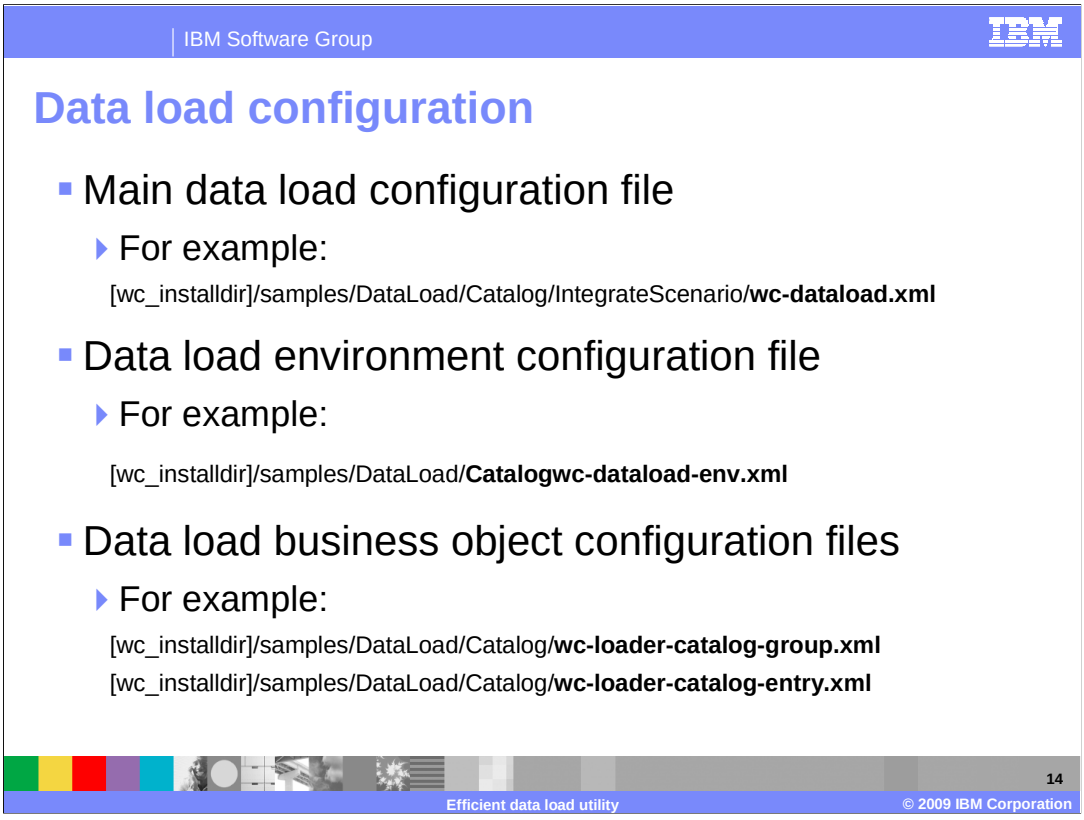


There are two scenarios for data loading.

If it is your first time loading data into the WebSphere Commerce database tables, this is called the **initial load**. For the initial load, you typically load a large amount of data. You can use JDBC load to write the data directly into the database. You can also use the native database load to generate the native database loadable files, then call the database load utilities to load these files. To get better performance, you typically use the Insert data load mode and specify the key range.

The database native load bypasses the database triggers. If you have a staging server environment, you need to load these files to both staging and production servers.

Delta load is used to update your existing data in the database. For this scenario, you typically use the Replace mode and use the JDBC load. You can load the data either on your staging server or production server.

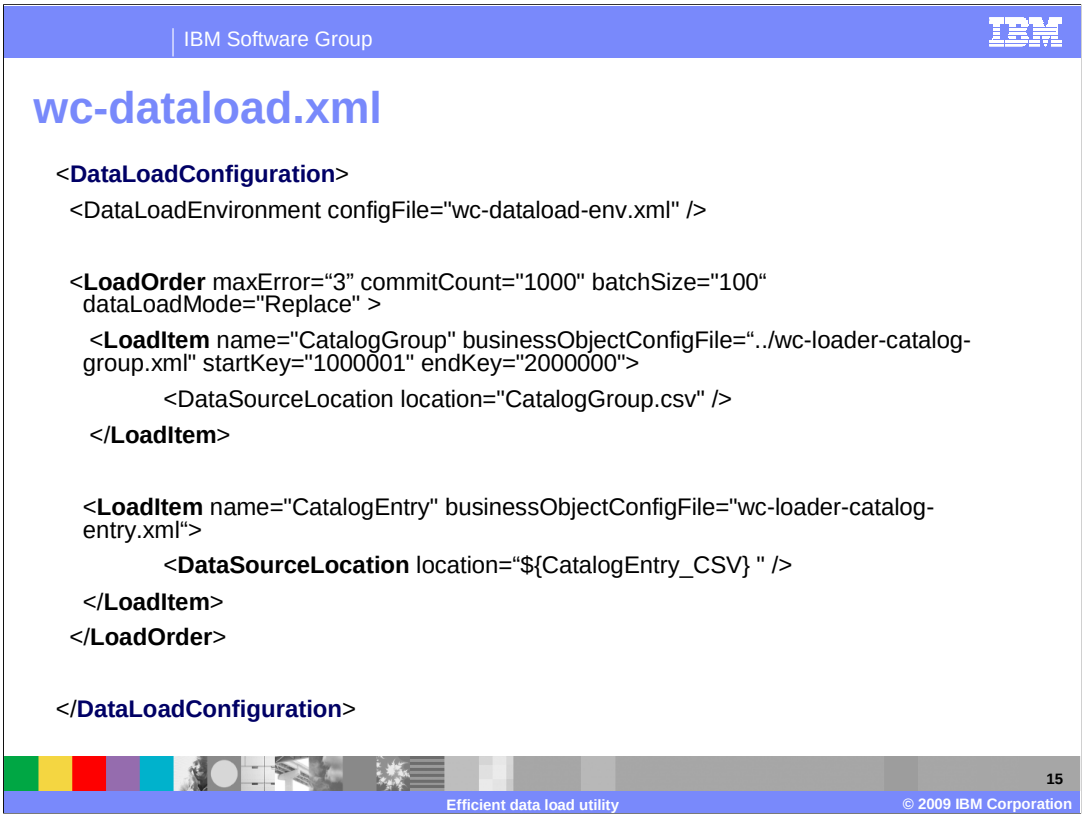


Data load execution is controlled by XML configuration files.

The **main data load configuration file** defines the execution flow. In this configuration file you can define what business object you want to load, where to find the business object configuration file and the data input file for this business object.

The **data load environment configuration file** defines global properties. It defines the database connection information such as database name, location, user ID and password. It also contains the information which is used by the business context component, such as language, currency and store ID.

The **data load business object configuration file** defines the mapping between the CSV file columns and the business object. Typically, one type of CSV file will have one business object configuration file. In one data load run, you can define multiple data load business object configuration files.



This slide shows an example of the main data load configuration file.

This file points to other configuration files, such as the data load environment file, data load configuration files, and input data source files. The main data load configuration file contains a list of the load items to be loaded. For the example in the slide, the data load first loads the catalog group data, and then loads the catalog entries data.

Attributes for element <LoadOrder>

- maxError
  - 1: default value
  - 0: do not stop for errors
- commitCount
  - 1: default value
  - 0: do not commit until the this load item finishes
- batchSize
  - 1: default value
  - 0: batches for whole load
- dataLoadMode
  - Insert: used for insert in the initial data load
  - Delete: used for delete
  - Replace: used for insert, delete and update

The attribute **maxError** is maximum number of acceptable errors. If an error occurs while loading and the number of errors is less than the maxError, the data load will continue to execute. When the error number reaches the maxError, the data load is terminated. If the maxError is not specified, the default value is 1. If the maxError is 0, it means all errors are ignored and the program continues to run to the end.

The attribute **commitCount** controls the number of records to be committed to a database in a transaction. The default is 1. If the commitCount is 0, it means it will not commit until this load item finishes.

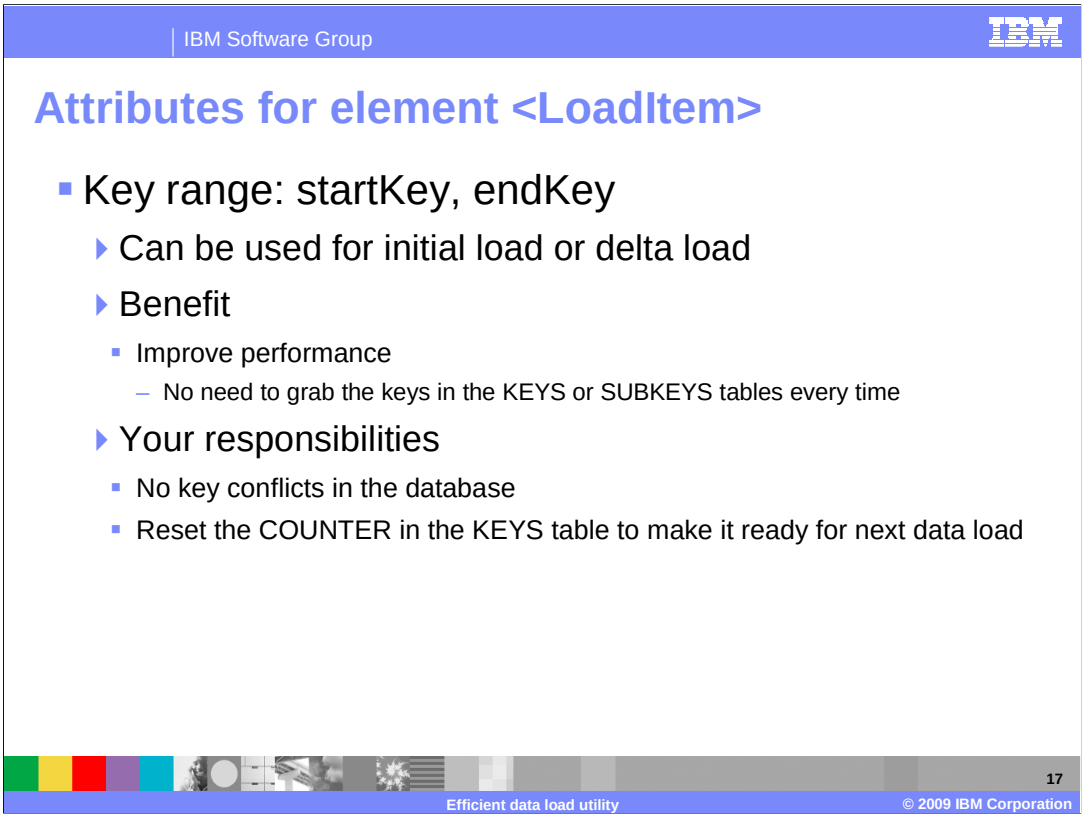
batchSize is an optional attribute which specifies how many lines of records to process before calling the JDBC batch update. The default is 1, which means it doesn't use JDBC batch update. If the batchSize is 0, it means it batches for the whole load.

The **dataLoadMode** attribute can have a value of Insert, Delete, or Replace.

Insert is used if everything in the CSV file will be inserted in the database. This mode is typically used in the initial data load.

Delete is used if everything in the CSV file will be deleted from the database.

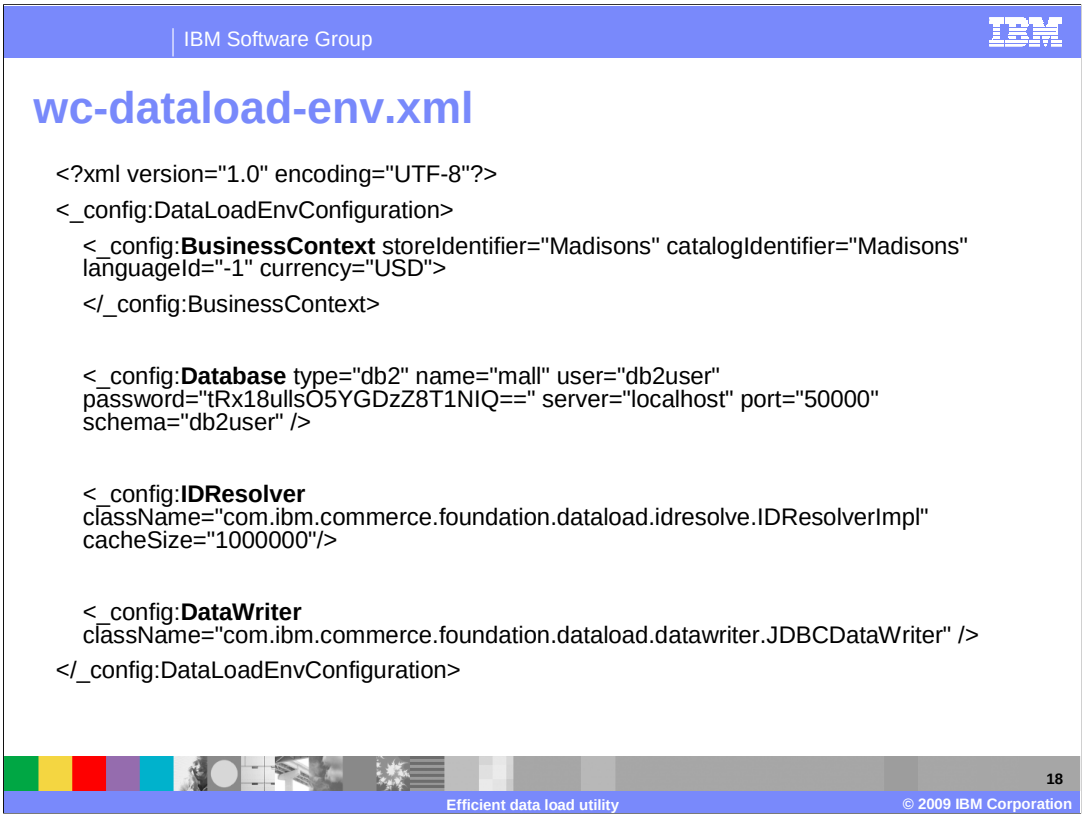
Replace is used if everything in the CSV file will be updated to the database. For example, if one line in the CSV file represents a new object, it will be inserted. If the object is in the database already, it will be updated. If there is a flag in the CSV file to indicate this object should be deleted, it will be deleted.



You can use startKey and endKey to define your key range.

The fixed key range can be used by an initial load or delta load to improve the performance. When the key range attributes are used, the code does not need to grab the keys in the KEYS or SUBKEYS tables every time when it needs more keys. You are responsible for making sure there are no conflicting keys in the database. After the data load finishes, you are responsible for resetting the COUNTER in the KEYS table to ensure that the next data load run uses the correct starting key.

COUNTER is the next starting key value to be fetched by the server from the database.



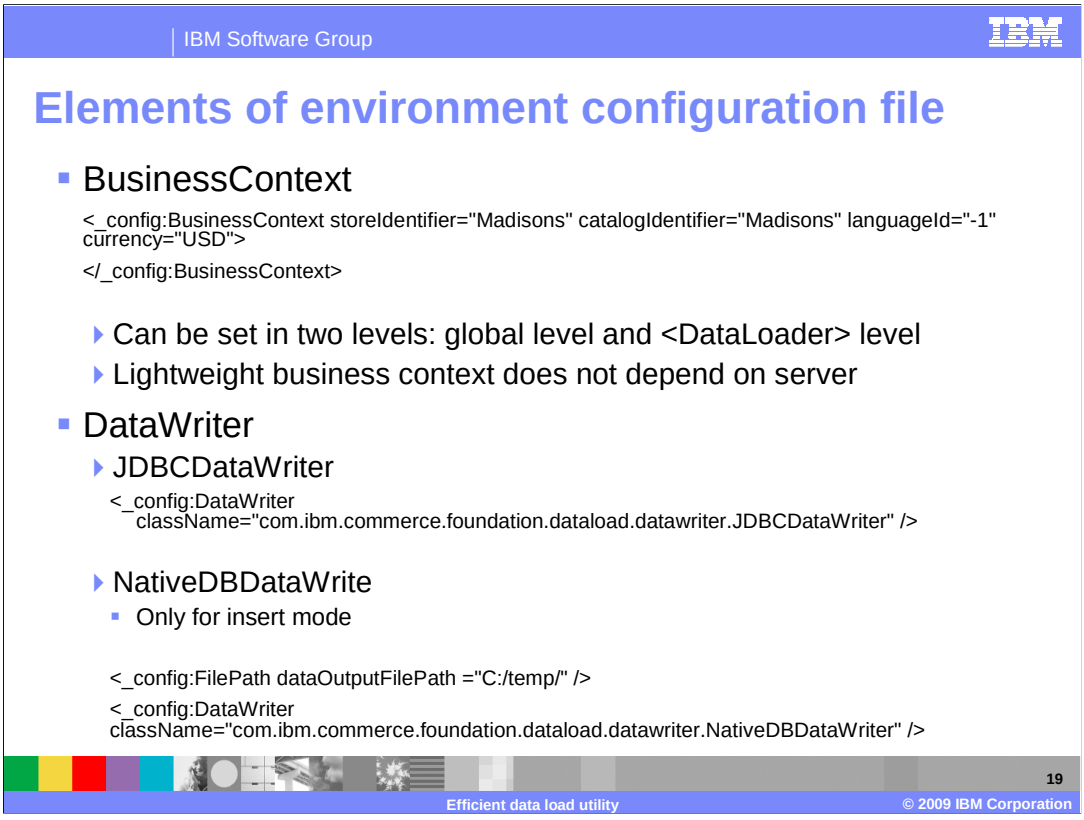
This slide shows an example of the data load environment configuration file.

This file contains the environment information and global business context properties. It specifies the properties for the database connection, global properties used by the business context component, and the class names for the ID resolver and Data Writer.

The user ID for the database connection is the one that you specified when you created the WebSphere Commerce instance. The password for this ID is encrypted.

Typically you don't need to change the IDResolver className. You can specify cacheSize for the number of resolved keys in memory.

For initial load, you should specify a large ID resolver cache size. For delta load and a large database, you should set the cache size to 0.



BusinessContext specifies default business context information, such as storeId and languageId.

Business context data can be set at two levels. One is at a global level in the wc-dataload-env.xml file and the other is in each <DataLoader> element. If a different business context is defined at the global level and the DataLoader level, the one defined at the DataLoader level will override the one defined at the global level.

The business context used by data load has been redesigned. It is lightweight enough so that it can be used by an offline data load utility. That is, the server does not need to be running to use the data load utility.

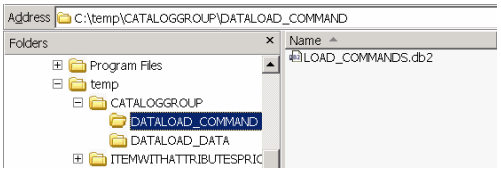
Two **DataWriter** types are provided in V7, **JDBCDataWriter** and **NativeDBDataWriter**.

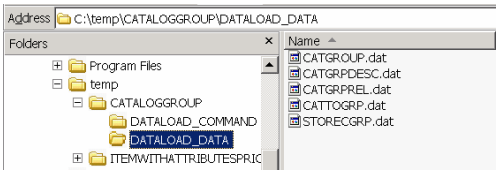
If the NativeDBDataWriter is used, it will generate multiple native loadable files, one file for each database table. You can specify the output files location in <FilePath>. A DBA can run the database-specific native load utility to load these native loadable files into the database.

The NativeDBDataWriter can only be used in **insert** mode. You need to make sure there are no duplicate records in the database.

Native DB data writer

- Load item names are used for subdirectory names
- For each load item
  - DATALOAD_DATA
  - DATALOAD_COMMAND
- Limitation
  - You might not be able to run multiple load items in one pass





The NativeDBDataWriter will produce multiple delimited output data files, with each file having records for one database table. Along with the delimited data files, the writer will also produce a command file which will contain the commands to load the generated data files to the database using the database native load utilities.

In this example, the directory c:/temp/ is the root output directory. Under this root directory, some subdirectories with the load item names are created. That is, all files generated by the native load are grouped by the load item name. Under each load item name directory, two sub-directories are created:

The DATALOAD_DATA directory is for **all output data files.**

The DATALOAD_COMMAND directory is for the **native db command file.**

There is a limitation in that you might not be able to run multiple load items in one pass. If one load item has a dependence on another load item, for example catalog entries are dependent on catalog groups and then you can not run the catalog entry data load without saving the dependent catalog groups to the database first. Otherwise, the ID resolver will have problems resolving the ID.

wc-loader-catalog-group.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<DataloadBusinessObjectConfiguration>
<DataLoader className="com.ibm.commerce.foundation.dataload.BusinessObjectLoader">
 <DataReader className="com.ibm.commerce.foundation.dataload.datareader.CSVReader" firstLineIsHeader="true"
    useHeaderAsColumnName="true" />
 <BusinessObjectBuilder className="com.ibm.commerce.foundation.dataload.businessobjectbuilder.BaseBusinessObjectBuilder"
    packageName="com.ibm.commerce.catalog.facade.datatypes.CatalogPackage" dataObjectType="CatalogGroupType"  >
  <DataMapping>
   < mapping xpath="CatalogGroupIdentifier/ExternalIdentifier/GroupIdentifier" value="GroupIdentifier" />
   <mapping xpath="ParentCatalogGroupIdentifier/ExternalIdentifier/GroupIdentifier" value="ParentGroupIdentifier" />
   <mapping xpath="topCatalogGroup" value="TopGroup" />
   <mapping xpath="displaySequence" value="Sequence" />
   <mapping xpath="Description[0]/Name" value="Name" />
   <mapping xpath="Description[0]/ShortDescription" value="ShortDescription"   />
    <mapping xpath="Description[0]/Attributes/published" value="Published" />
    <mapping xpath="Description[0]/Attributes/note" value="Note" />
   <mapping xpath="Attributes/field1" value="Field1" />
    <mapping xpath="" value="Delete" deleteValue="1"/>
  </DataMapping>
  <BusinessObjectMediator className="com.ibm.commerce.catalog.dataload.mediator.CatalogGroupMediator"
     componentId="com.ibm.commerce.catalog"/>
 </BusinessObjectBuilder>
</DataLoader>
</DataloadBusinessObjectConfiguration>
```


You can have many loader configuration files in one data load run. The loader configuration file is used to define the data load for a specific WebSphere Commerce business object.

In this configuration file, you specify the data reader class, the business object builder class, the mapping between the input file columns and the business object, and the business object mediator.

This configuration file allows you to use your own data reader class if you want to read the data from other data sources, for example a database table or XML file.

You can also specify your own customized business object mediator class in this configuration file.

Mapping between XPath and CSV file column

CatalogGroups.csv

GroupIdentifier	TopGroup	ParentGroupIdentifier	Sequence	Name	ShortDescription	LongDescription	Thumbnail	FullImage	Keyword	Delete
Mens Fashions	TRUE		1	Men's	All men's dressw	Find the latest in	images/cat	images/ca	mens fash	0
Womens Fashi	TRUE		2	Wome	All women's dres	Find the latest in	images/cat	images/ca	womens fa	0
Pants		Mens Fashions	1	Pants	Denim, cargo an	Find the perfect	images/cat	images/ca	mens fash	0
Shirts		Mens Fashions	2	Shirts	Casual, dress an	Find the perfect	images/cat	images/ca	mens fash	0
Activewear		Womens Fashions	1	Active	Activewear for m	Fitness clothing	images/cat	images/ca	womens fashions,	
Accessory		Womens Fashions	2	Acces	Accessories for	Find all accesso	images/cat	images/ca	womens fashions,	

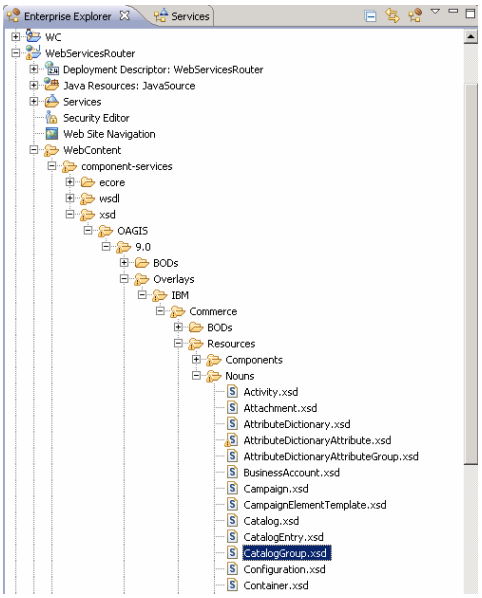
wc-loader-catalog-group.xml

```
<DataMapping>
 < mapping xpath="CatalogGroupIdentifier/ExternalIdentifier/GroupIdentifier" value="GroupIdentifier" />
 <mapping xpath="ParentCatalogGroupIdentifier/ExternalIdentifier/GroupIdentifier" value="ParentGroupIdentifier" />
 <mapping xpath="topCatalogGroup" value="TopGroup" />
 <mapping xpath="displaySequence" value="Sequence" />
 <mapping xpath="Description[0]/Name" value="Name" />
 <mapping xpath="Description[0]/ShortDescription" value="ShortDescription"  />
 <mapping xpath="Description[0]/Attributes/published" value="Published" />
 <mapping xpath="Description[0]/Attributes/note" value="Note" />
 <mapping xpath="Attributes/field1" value="Field1" />
 <mapping xpath="" value="Delete" deleteValue="1"/>
</DataMapping>
```


This slide shows an example of how XPath is mapped to a column in the CSV file. If the value in a mapping can not be found in the CSV file, a default value or NULL is used for this data load run.

Business object schema definition

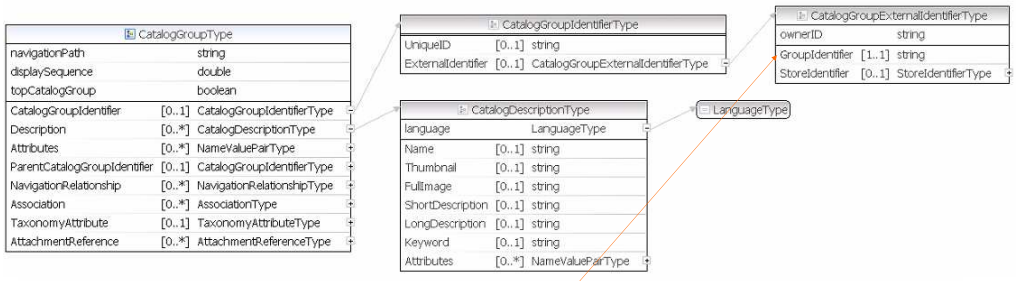
- View business object XML schema documents (XSD) in version 7 toolkit



You can view the business object schema definition directly from WebSphere Commerce Developer V7.

Follow the navigation path shown in the diagram to find the business object schema file. You can open the schema file in the XSD editor to view its source and its graphic structure.

View business object noun in XSD editor



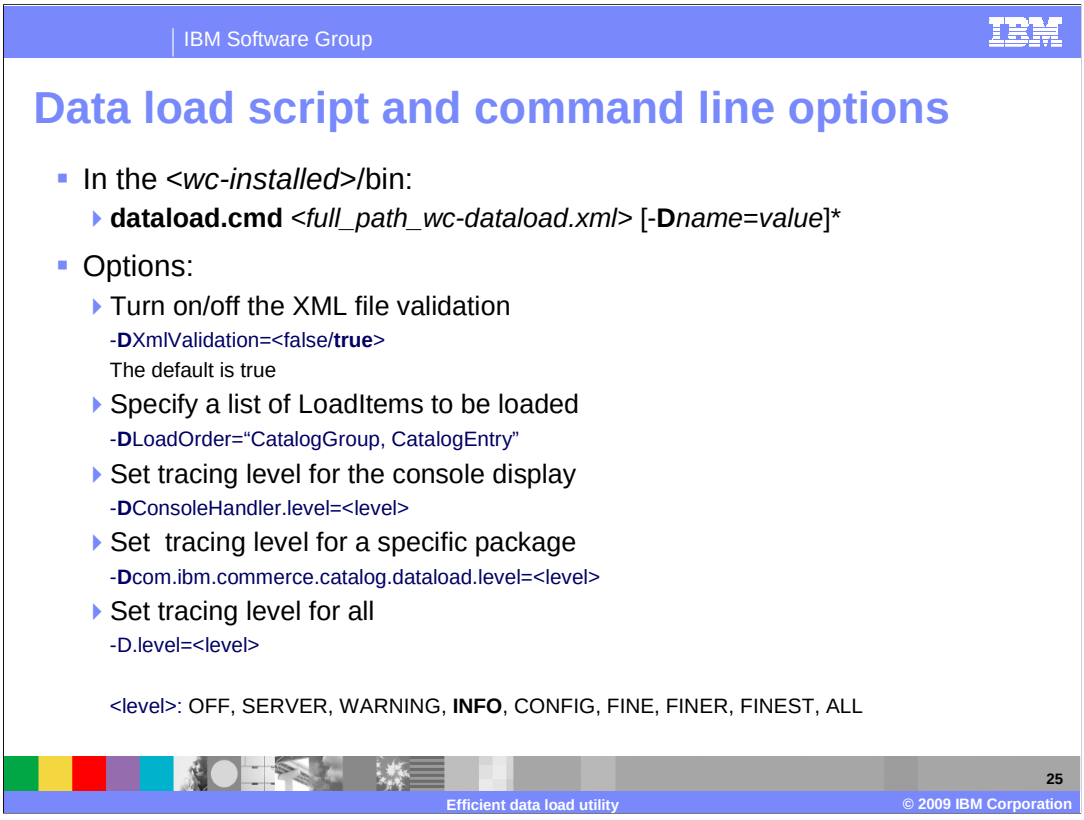

```
<DataMapping>
  < mapping xpath="CatalogGroupIdentifier/ExternalIdentifier/GroupIdentifier" value="GroupIdentifier" />
  <mapping xpath="ParentCatalogGroupIdentifier/ExternalIdentifier/GroupIdentifier" value="ParentGroupIdentifier" />
  <mapping xpath="topCatalogGroup" value="TopGroup" />
  <mapping xpath="displaySequence" value="Sequence" />
  <mapping xpath="Description[0]/Name" value="Name" />
  <mapping xpath="Description[0]/ShortDescription" value="ShortDescription"  />
  <mapping xpath="Description[0]/Attributes/published" value="Published" />
  <mapping xpath="Description[0]/Attributes/note" value="Note" />
  <mapping xpath="Attributes/field1" value="Field1" />
  <mapping xpath="" value="Delete" deleteValue="1"/>
</DataMapping>
```


This slide shows an example of the business object schema as viewed from the XSD editor.

After you open the business object schema in the XSD editor view, you can see the data elements in the schema.

In the graphical schema diagram, you can click the plus (+) sign and navigate the element tree.

The XPath expression matches the structure of the logical schema. The CSV column in the input file should be mapped to a primitive XML schema type, such as Boolean.



A command file is provided for the data load utility. It takes one mandatory parameter, the path to the main data load configuration file, and several optional name-value pairs.

You can use the optional parameters to turn on or off the XML validation and set the trace level at several different granularities.

If you do not want to load all items listed in the wc-dataload.xml, you can list what items you want to load in the command line using –DLoadOrder option.

Variable substitution in configuration XML files

- Variable substitution
  - wc-dataload.xml

```
<LoadItem name="CatalogEntry" businessObjectConfigFile="wc-
  loader-catalog-entry.xml“>
    <DataSourceLocation location=“${CatalogEntry_CSV} " />
  </LoadItem>
```

  - Command line

```
Dataload.cmd wc-dataload.xml -
  DCatalogEntry_CSV=CatalogEntry.csv
```


In the data load configuration XML files, you can use variables in the attribute values. The variable has the format ${name}.

The real values in the configuration XML file are resolved at the time data load utility is running. Values can be passed in from the command line as shown in the slide.

Sample output messages in the console window

Load summary for load item: ItemWithAttributesPrice.

Business Object Configuration: **wc-loader-item-attributes-price.xml**
Data loader mode: Replace.
Batch size: 1.
Commit count: 100.
Error Tolerance Level: 1.
Error Count: 0.
Amount of data processed: 12.
Amount of business objects processed: 11.
Amount of business objects committed: 11.
Data loader initialization time: 0 seconds.
Data loader execution began: Mon Jun 29 15:11:00 CDT 2009
Data loader execution ended: Mon Jun 29 15:11:07 CDT 2009
Data loader completed in 6.625 seconds.
Total flush time: 0 seconds.
Total commit time: 0.016 seconds.
CSV file location: C:\IBM\WebSphere\CommerceServer70\samples\DataLoad\Catalog\IntegrateScenario\ItemsWithAttributesPrice.csv.
Affected tables (14):
Table name: CATGROUP, Affected number of rows: 0.

This slide shows part of the output message from the command console window.

Sample output messages in the console window

```
Table name: STORECGRP, Affected number of rows: 0.
Table name: CATGRPDESC, Affected number of rows: 0.
Table name: CATTOGRP, Affected number of rows: 0.
Table name: CATGRPREL, Affected number of rows: 0.
Table name: CATENTRY, Affected number of rows: 11.
Table name: STORECENT, Affected number of rows: 11.
Table name: CATENTDESC, Affected number of rows: 11.
Table name: CATGPENREL, Affected number of rows: 11.
Table name: CATENTREL, Affected number of rows: 8.
Table name: LISTPRICE, Affected number of rows: 11.
Table name: ATTRVALUE, Affected number of rows: 16.
Table name: OFFER, Affected number of rows: 11.
Table name: OFFERPRICE, Affected number of rows: 11.
---------------------------------------------------------------------------
Jun 29, 2009 3:11:07 PM com.ibm.commerce.foundation.dataload.DataLoaderMain logExitCode
INFO:
Program exiting with exit code: 0.
Load completed sucessfully with no errors.
Jun 29, 2009 3:11:07 PM com.ibm.commerce.foundation.dataload.DataLoaderMain logEndDateAndTime
INFO: Load ended at: Mon Jun 29 15:11:07 CDT 2009
Jun 29, 2009 3:11:07 PM com.ibm.commerce.foundation.dataload.DataLoaderMain logEndDateAndTime
INFO: Load completed in 24.562 seconds.
```


This slide shows another part of the output message from the command console window.

If the data load finishes successfully without any errors, the system exit value is 0. If the dataload runs to the end with errors the system exit value is 1. As described on an earlier slide, the data load utility can continue to run after an error occurs as long as the number of errors is below the maxError threshold.

Problem determination

- Two types of log file
  - Log for all load items

    [WC_HOME]/logs/wc-dataload.log
  - Error log for each load item if error occurred during the load

    [WC_HOME]/logs/[load-item-name]_ERROR_[time-stamp].log
- Data load trace level

  dataload ..\samples\DataLoad\Catalog\IntegrateScenario\wc-dataload.xml -D.level=FINER

The wc-dataload.log file is generated each time the data load utility is run. A separate error log is generated for each load item where an error occurs. If no errors occur, then no error logs are generated.

By default, the trace level is **INFO**. You can turn on a specific data load package trace or use the level flag to set the trace level for all packages.

Load custom extension tables with UserData

- Information Center tutorial mapping the custom extension tables with the UserData element:
  - Create custom tables XWARRANTY and XCAREINSTRUCTION in database:
    http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.soa.doc/tutorial/twvfounduserdata1.htm
  - Run DSL wizard in the WebSphere Commerce Developer:
    http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.soa.doc/tutorial/twvfounduserdata2.htm
- Deploy the generated assets:
  - Package the physical SDOs for the custom tables into a jar **catalog-ext.jar** and deploy it to: <WC_installed>/ext/lib directory.
  - Deploy the generated custom extension configuration directory with files **com.ibm.commerce.catalog-ext** to: <WC_installed>/instances/<instance_name>/xml/config directory.
- In the wc-loader-catalog-entry.xml, add the corresponding mapping:

```
<_config:mapping xpath="UserData/UserDataField/warterm" value="WarrantyTerm" />
<_config:mapping xpath="UserData/UserDataField/wartype" value="WarrantyType" />
<_config:mapping xpath="Description/Attributes/careinstruction" value="CareInstruction"  />
```


The data load **framework** is designed to be customizable. You can customize any layer in the data load configuration. The most common customizations are custom data readers and custom business object mediators.

Besides these data load framework related customizations, if you extend the WebSphere Commerce schema with your own tables, you need to customize the business object layer. You need to use the Data Service Layer Wizard to generate object-relational metadata and physical data objects.

Directly loading the data into tables

- Difference between the data loading by business object and by table

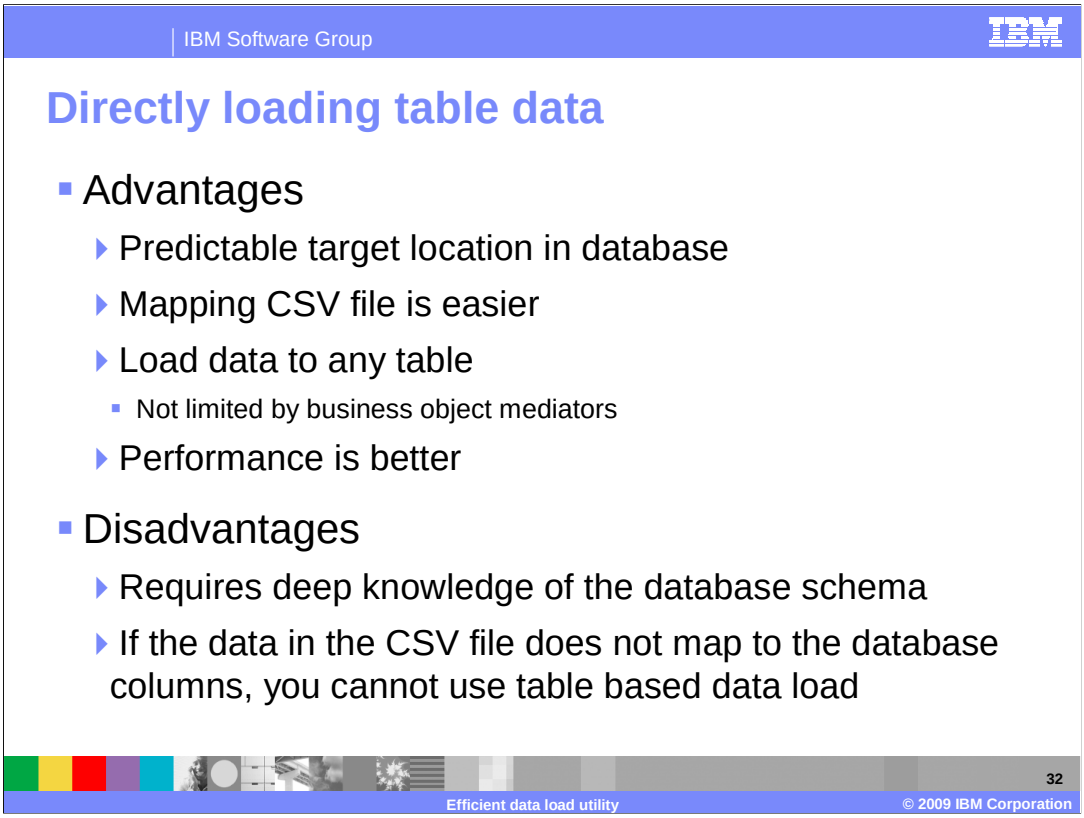
Loading by business object	Loading by table
Map CSV columns to the XPath of the business object	Map the CSV columns to the database table columns
Business object is built by BusinessObjectBuilder	**TableObject** is built by TableObjectBuilder
BusinessObjectMediator transfer business object into physical object	**TableObjectMediator** transfers TableObject into physical objects

- Loader configuration file

```
<_config:DataLoader className="com.ibm.commerce.foundation.dataload.BusinessObjectLoader">
  <_config:DataReader className="com.ibm.commerce.foundation.dataload.datareader.CSVReader" firstLineIsHeader="true" >
  </_config:DataReader>
  <_config:BusinessObjectBuilder className="com.ibm.commerce.foundation.dataload.businessobjectbuilder.TableObjectBuilder" >
    .....
    <_config:Table name="USERREG">
    <_config:Column name="USERS_ID" value="MEMBER_ID" valueFrom="IDResolve" />
      <_config:Column name="LOGONID" value="email" />
      <_config:Column name="STATUS" value="1" valueFrom="Fixed" />
      <_config:Column name="PLCYACCT_ID" value="-2" valueFrom="Fixed" />
      <_config:Column name="PASSWORDEXPIRED" value="0" valueFrom="Fixed" />
  </_config:Table>
    .....
  <_config:BusinessObjectMediator className="com.ibm.commerce.foundation.dataload.businessobjectmediator.TableObjectMediator" >
    </_config:BusinessObjectMediator>
 </_config:BusinessObjectBuilder>
</_config:DataLoader>
```


The data load utility also supports loading the data into database tables directly.

To load a database table directly, you need to map the CSV columns to the database table columns in the loader configuration file. After the CSV reader reads a line of data, it returns a map. The map is passed to the TableObjectBuilder, where a TableObject is built. Then the TableObjectMediator is used to transform the TableObject to a physical object.



There are advantages to using direct database table loading.

The target locations of the data in the database are predictable. You know exactly how the physical tables and columns are populated.

Mapping the data in CSV file to table columns can be easier than mapping the data to a business object using XPath.

You can load data into any table. The business object based loading is limited to certain business objects.

Compared with business object based data loading, the direct database table loading is faster.

There are also disadvantages to directly database table loading.

First, you need a deep understanding of the database schema.

Second, if the data in the CSV file is not structured to match the database columns, you cannot use table based data loading.

Summary

- Five layers in data load framework
- Data load execution controlled by configuration XML files
- Three different kinds of configuration XML files
- Run data load on the command line with optional parameters
- Loading custom tables

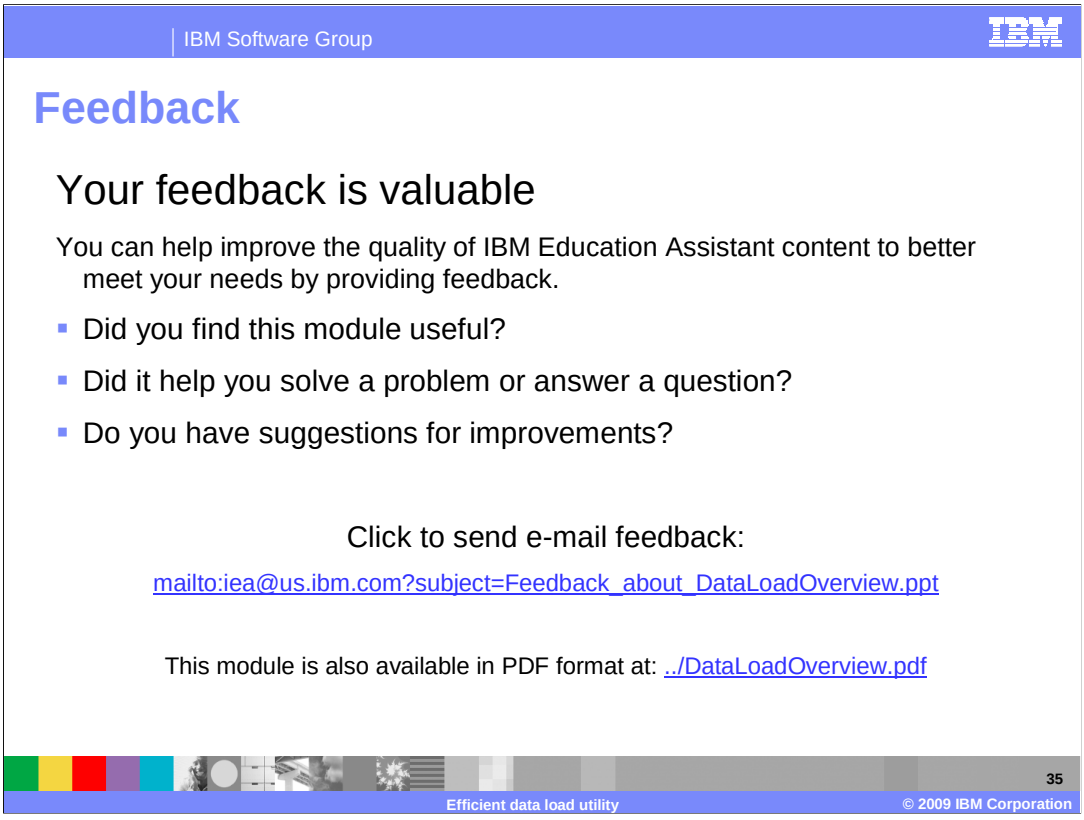
This presentation discussed the data load framework, the configuration files, command line invocation, and loading custom extension tables.

Reference

- Overview of mass load utilities in V6.0

  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/topic/com.ibm.commerce.data.doc/concepts/cmlpublishoverview.htm

This slide gives you a reference to how the mass loader utilities work in V6.0.



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

JDBC and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.