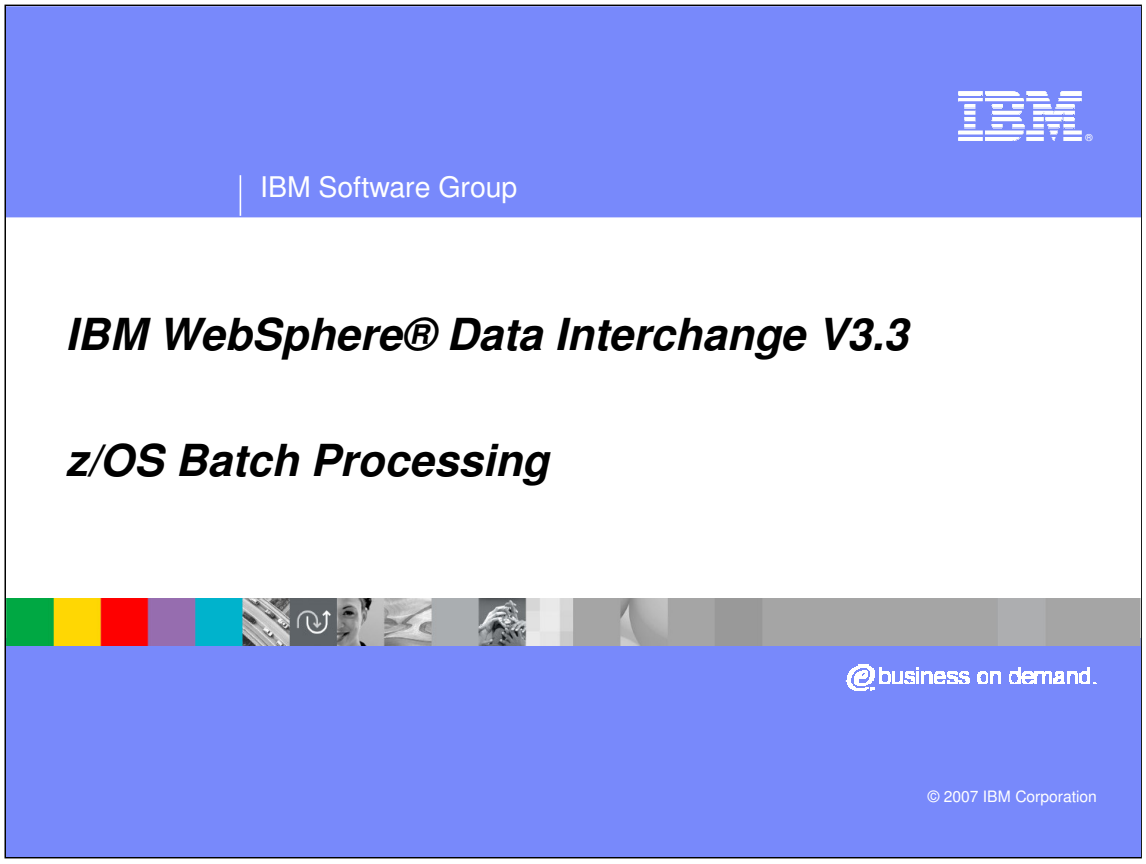


This presentation will describe the WebSphere Data Interchange Batch Processing Options.

Agenda

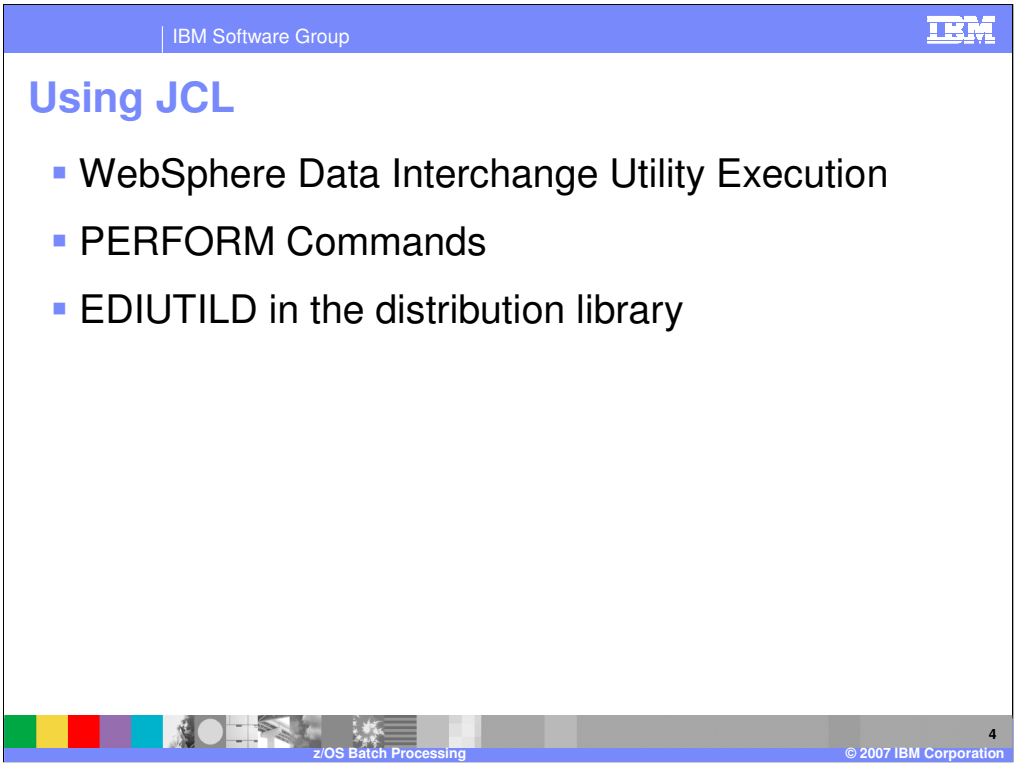
- Invoking WDI using JCL
- Using the WDI Translator API
- Reference

The presentation will describe the batch processing options available.

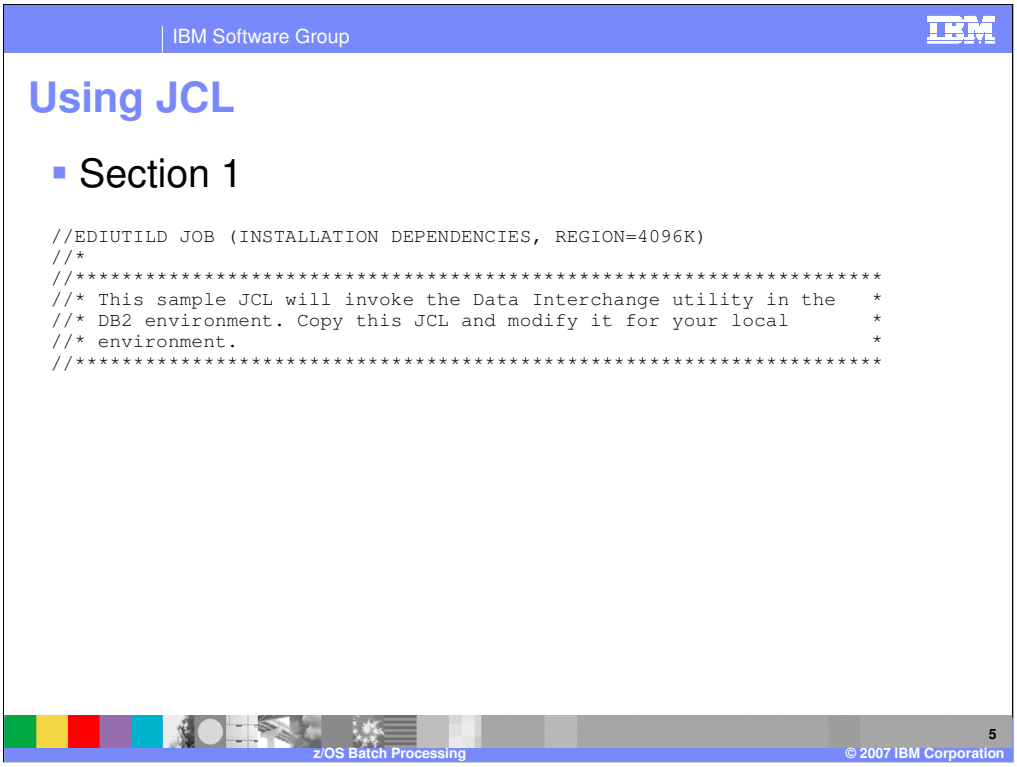
Section

Invoking WDI using JCL

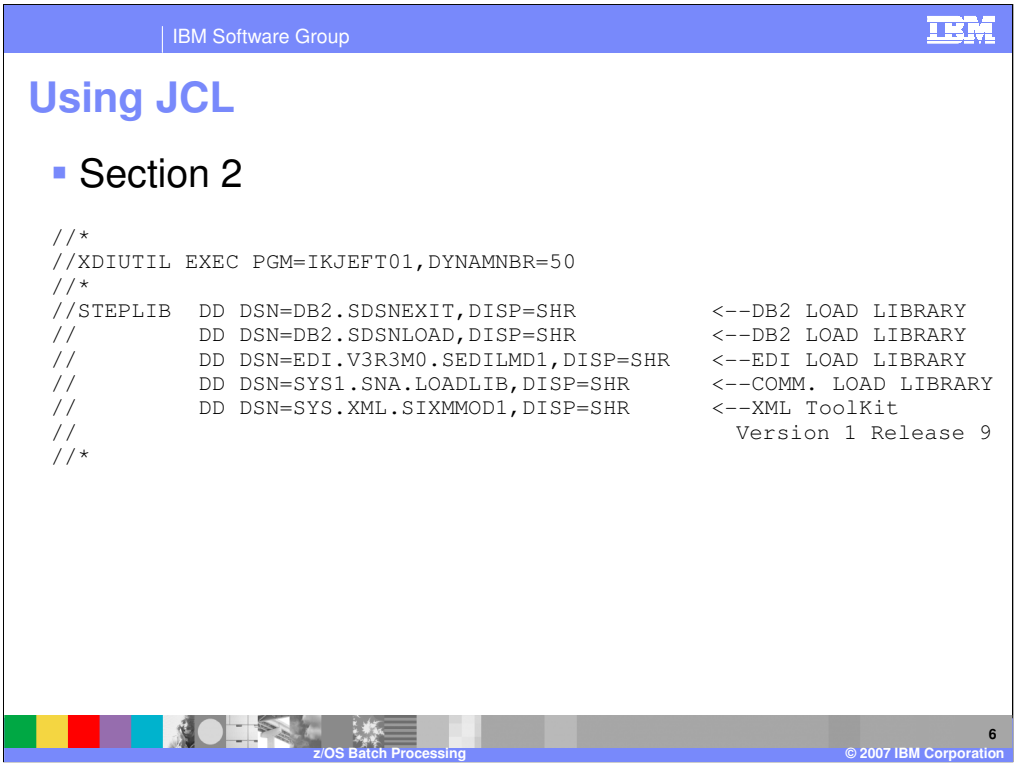
The Send Translation process.



Sample Utility JCLs are distributed in the JCL distribution library (EDI.V3R3M0.SEDIINS1). The DB2 version is in member EDIUTILD. The JCL is divided into subsections based primarily on function.

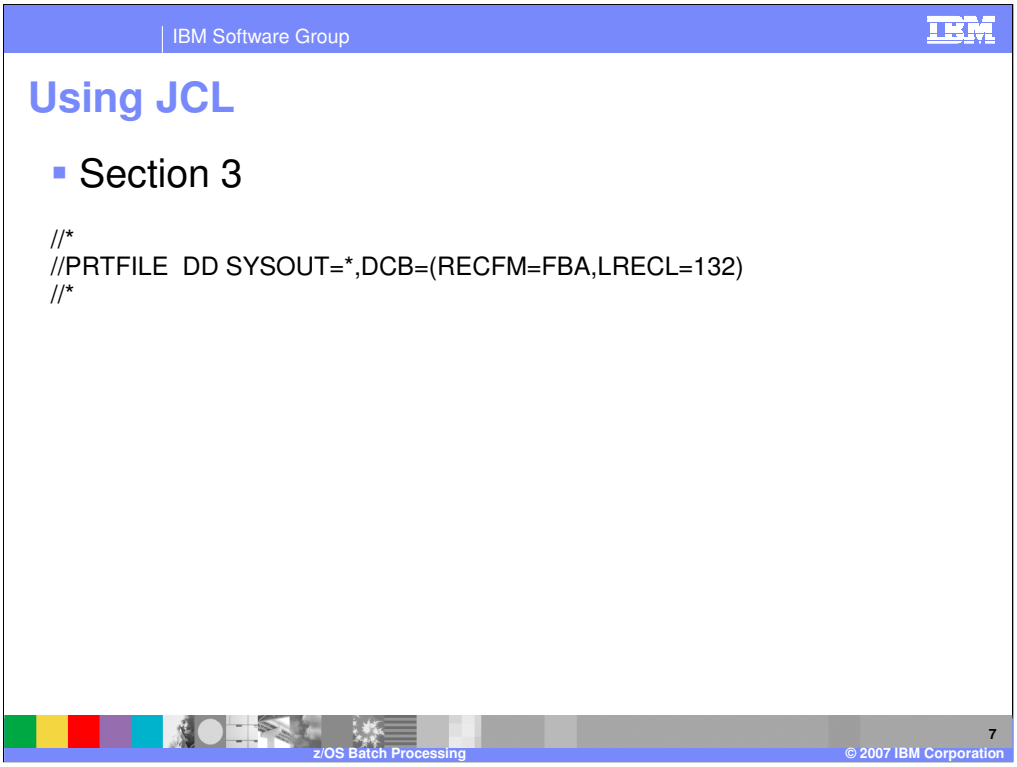


The beginning of the JCL states that certain elements of the JCL must be modified to run in your environment. This is usual and customary. The minimum region size to execute the WebSphere Data Interchange Utility is 4096 K.

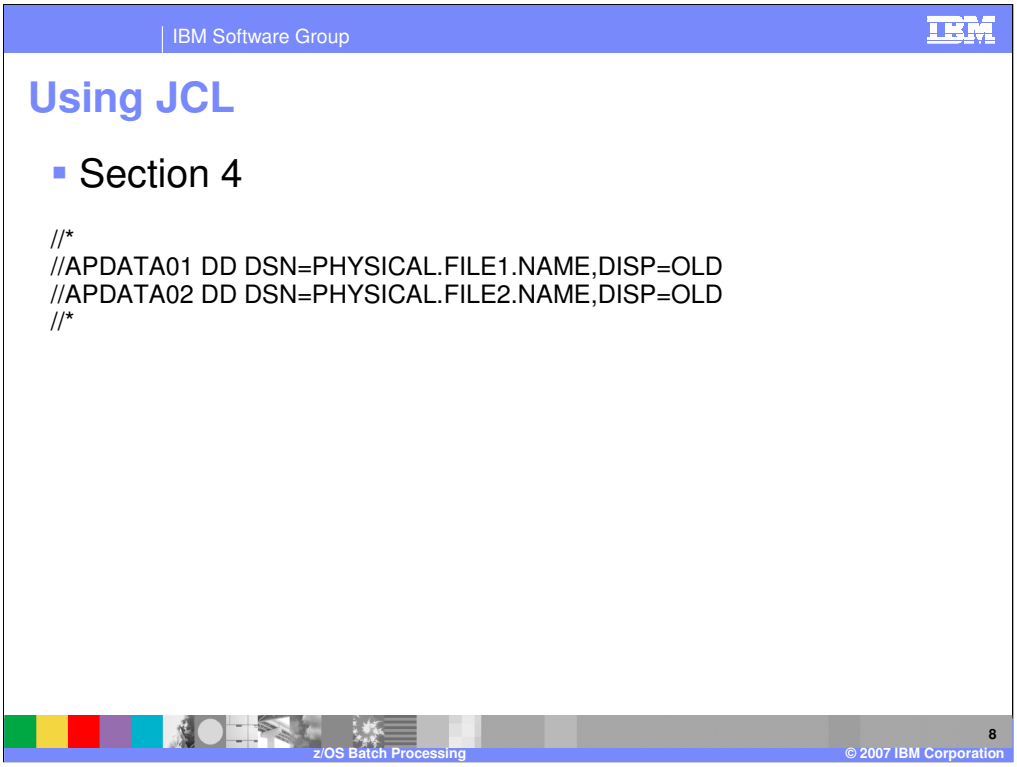


STEPLIB accesses the WebSphere Data Interchange load modules. If you make communication requests, STEPLIB must include the library where the communication routines reside. STEPLIB is not required if you have provided access to the necessary programs in other ways (such as adding the load libraries to the LNKLST, or loading all necessary programs into the link pack area).

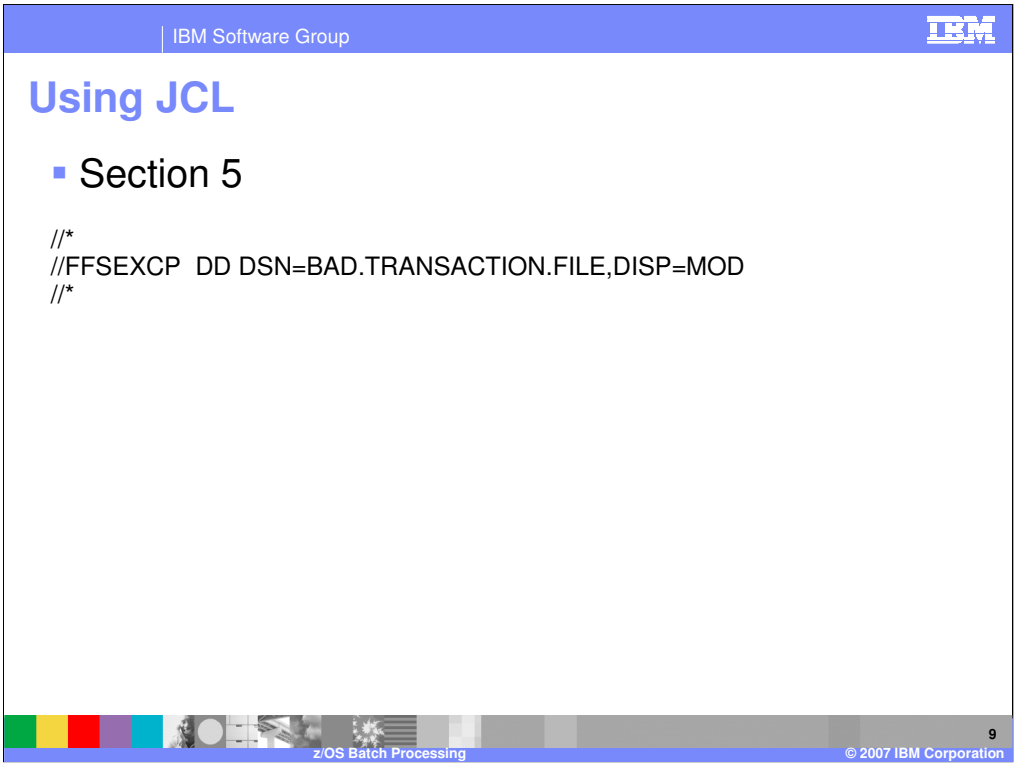
XML parameters should be added to the STEPLIB. The EDI.XML parameters (in the last line of code) should be edited as appropriate for your installation of XML Toolkit Version 1 Release 9, replacing SYS1 with the high level qualifier for your system. The XML Tool Kit is required for WebSphere Data Interchange data transformation processing.



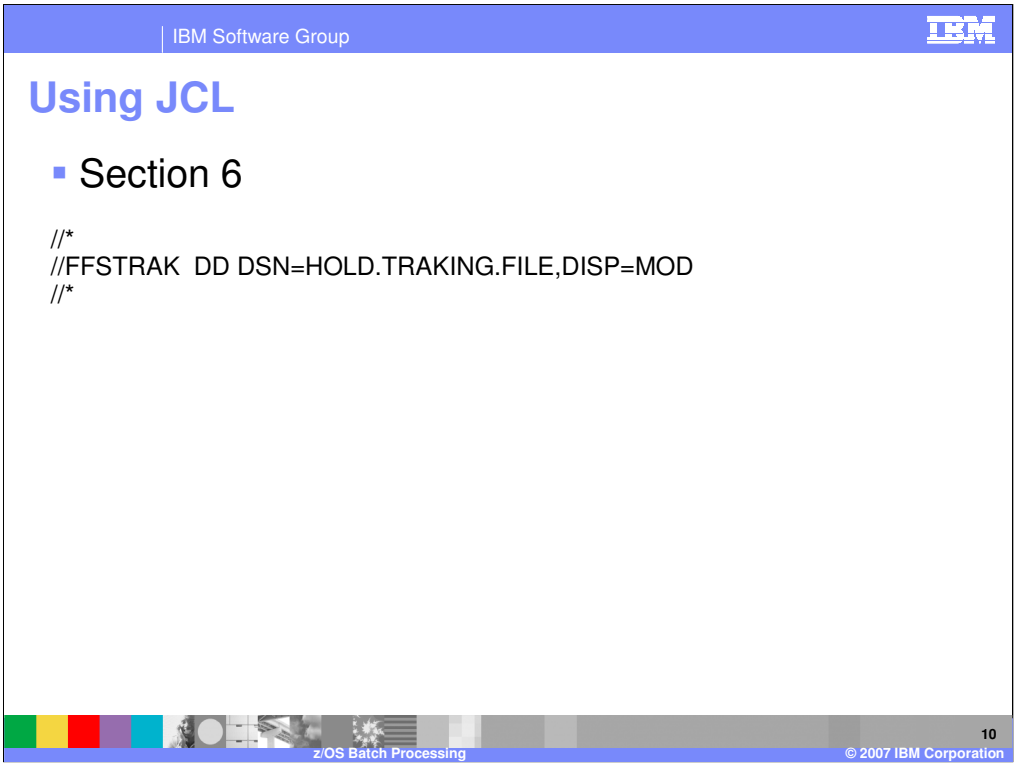
There are three print files in WebSphere Data Interchange. These are PRTFILE, XMLPRNT, and ADFPRNT. PRTFILE is required for all WebSphere Data Interchange Utility functions. XMLPRNT and ADFPRNT are optional. Error messages generated during function processing and a summary report are written to PRTFILE. Error messages are written to the optional files if specified. This data set contains fixed block addressing (FBA) or variable block addressing (VBA) records with a logical record length of 132.



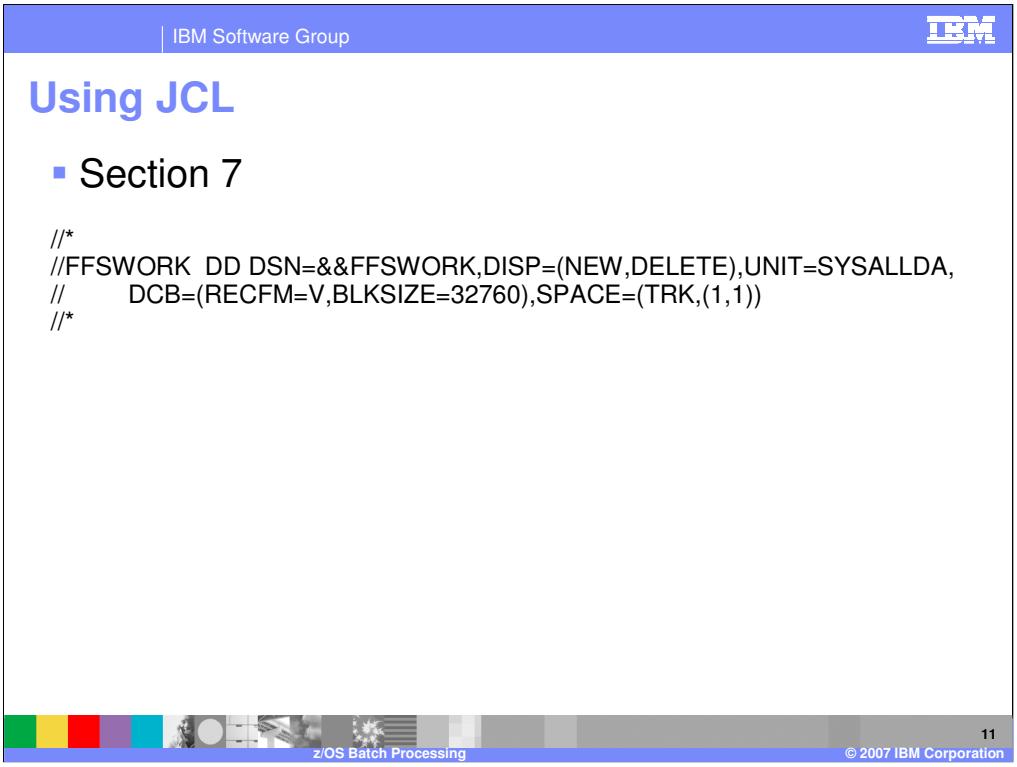
Translate to Standard files.. The APDATA01 and APDATA02 ddnames are examples of files containing application data that is to be translated into EDI standard format. The files can contain C and D records, or they can contain data in raw data format. The actual ddnames must match the names specified with the APPFILE keyword in the PERFORM command. A file is expected to contain C and D records unless the RAWFMTID keyword is provided. These data sets can contain fixed or variable length records.



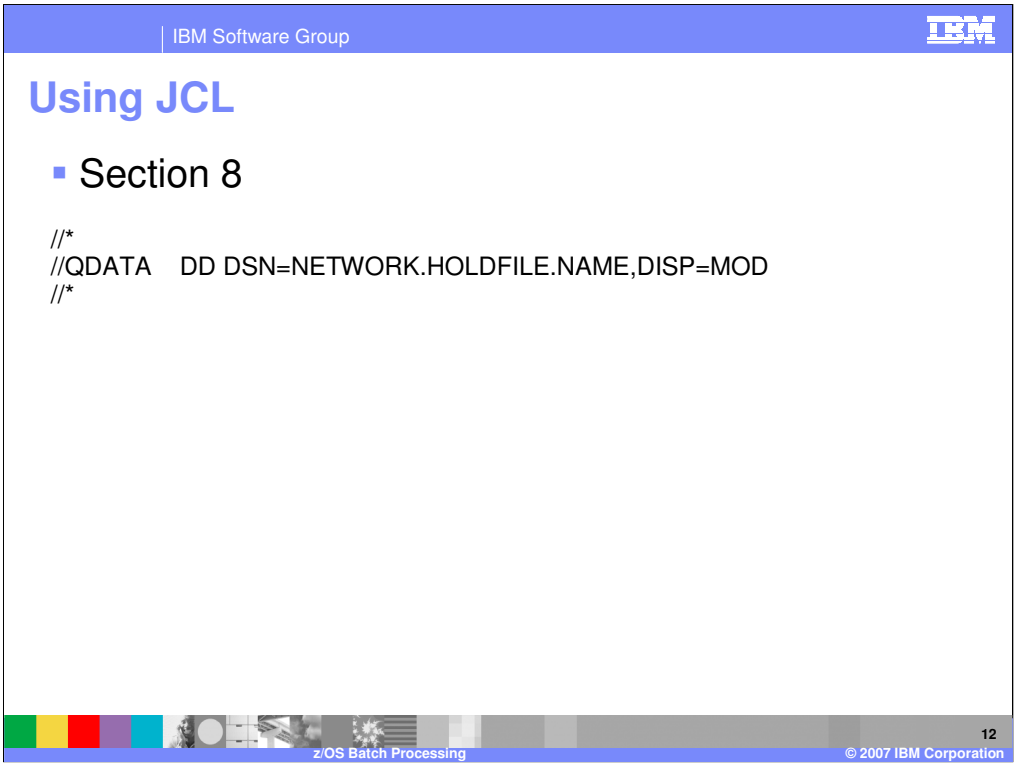
Destination files. When translating to EDI standard format, the WebSphere Data Interchange Utility writes transactions that were not translated successfully to FFSEXCP. When translating to application format, the Utility writes translated transactions to FFSEXCP if it can not write them to the intended file. (This is usually because the intended file could not be opened or is full). Optional records are also written to FFSEXCP if the tracking file (FFSTRAK) does not exist and your application data is in C and D format. FFSEXCP must be large enough to contain the largest data record you are translating, and the largest information record the translator might return. Use the JCL DISP option to control whether FFSEXCP is cleared or appended to during the first use. After the first use, WebSphere Data Interchange automatically appends to the file. FFSEXCP can contain fixed or variable length records.



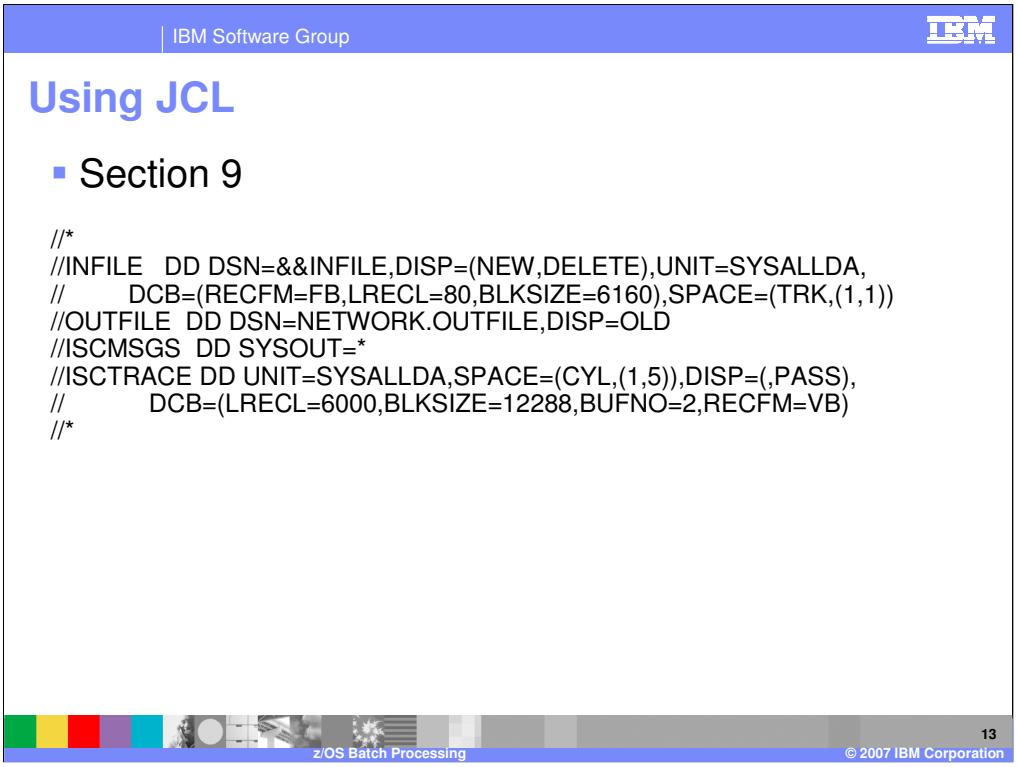
FFSTRAK holds the optional information records if they are requested. If you are using C and D records and if FFSTRAK is not provided, the optional records are written to the exception file (FFSEXCP). If your application data is in raw data format and FFSTRAK is not provided, no optional records are written even if they are requested. FFSTRAK must be large enough to accommodate the largest information record. FFSTRAK can contain fixed or variable length records.



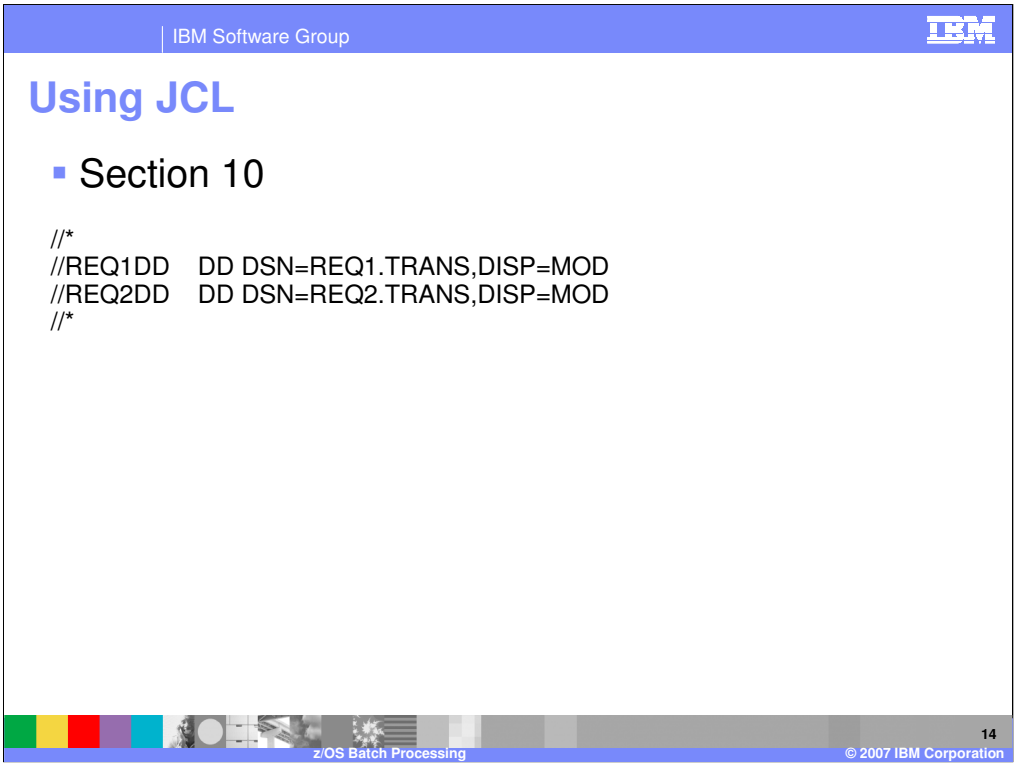
FFSWORK is required only when translating from application format to EDI standard format. FFSWORK holds the current transaction in case of translation errors. If an error occurs, the current transaction is copied from the FFSWORK file to the exception file (FFSEXCP). This data set can contain variable record format with a logical record length of 32756. If your system guidelines allow, specifying UNIT=VIO on this DD statement will drastically reduce import/export exceptions on the data set. Allocating with a variable blocked (VB) record format will also reduce the number of I/O EXCPs on the data set.



Envelope data file. If your Utility request involves enveloping and/or sending of transaction data, this is the section of JCL that defines the EDI standard data file to envelope into and send from. The ddname should match one of the following: v The Trans data queue field value as specified in the Network Profile (NETPROF). If a name has not been supplied in the Network Profile member, the default is **QDATA**. The ddname specified holds transactions that have ISA, ICS, or GS enveloping. The same file name with an **E** appended (such as QDATAE) holds transactions that have either UNB or STX enveloping. The same file name with a **U** appended (such as QDATAU) holds transactions that have BG enveloping. There must be a ddname specified for each network that can be accessed during the run. v The value specified in the FILEID keyword on the PERFORM command. If this keyword is specified, that file is used to hold all envelope types for all networks. This data set can contain fixed-length or variable-length records with a logical record length of 80 or greater.



Network communications files. This group of JCL statements is used when sending to or receiving from a network. The JCL shown here is required for the AT&T Global Network. Each network has its own JCL requirements.

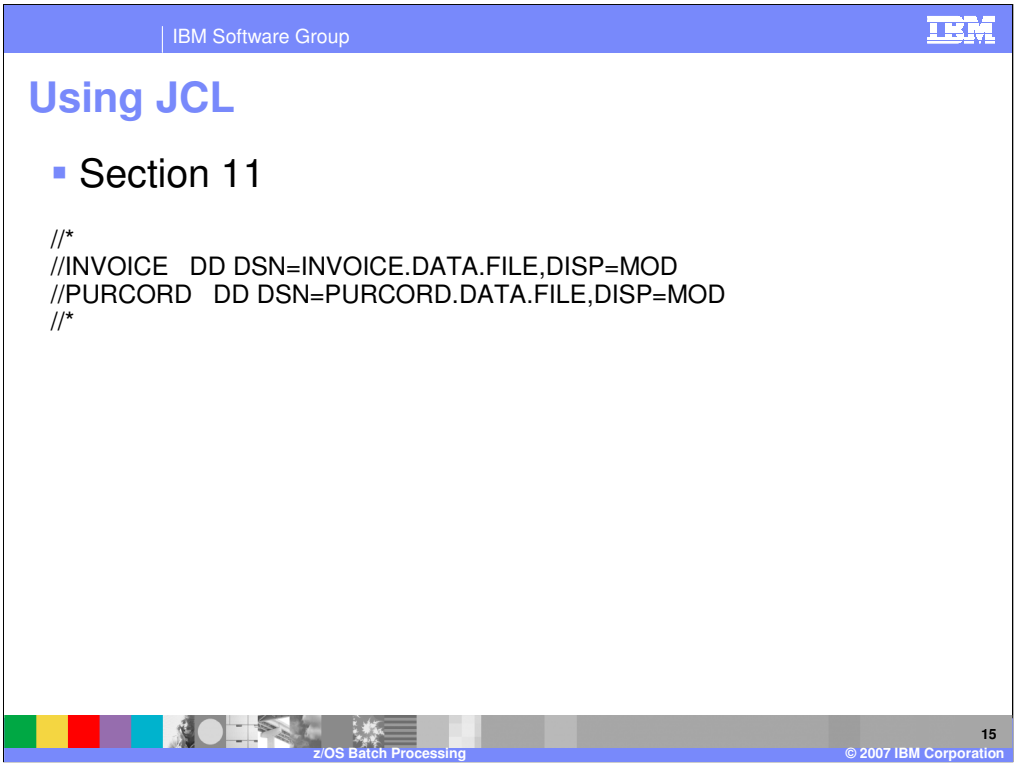


EDI Standards files. For receiving and deenveloping transaction data, this section of JCL defines the EDI standard data file to receive into and deenvelope from. The ddname should match one of the following:

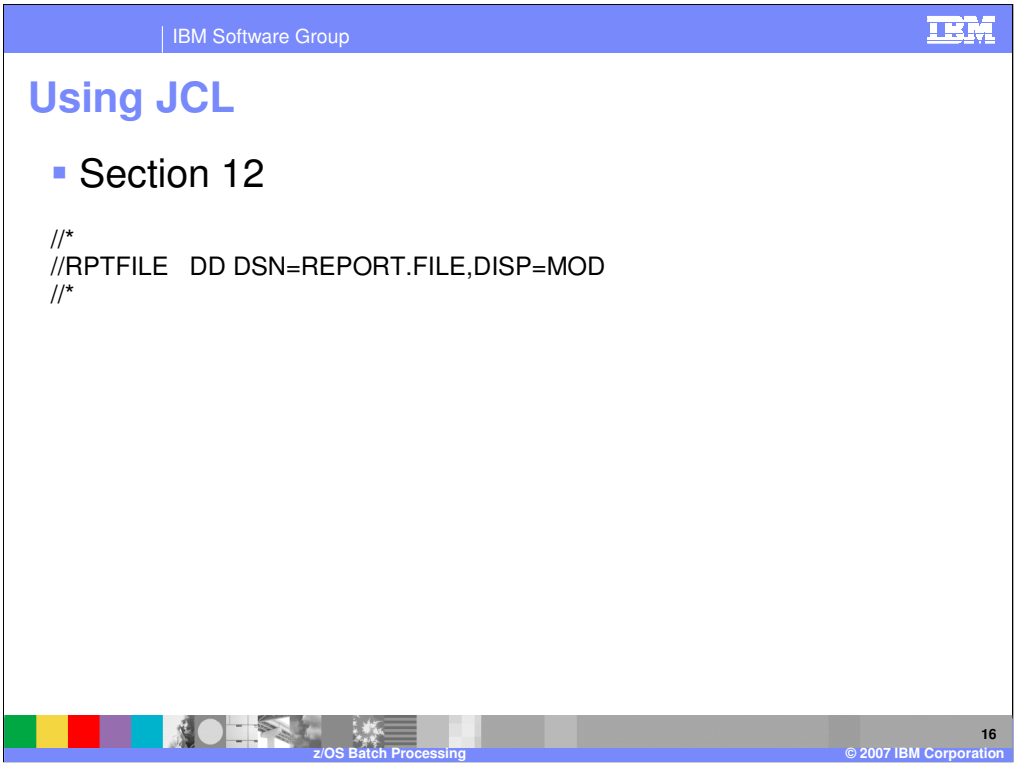
-The Receive file name field value as specified in the mailbox (requestor) profile (REQPROF) associated with the REQID keyword in the PERFORM command. There must be a ddname here for each requestor that can be processed during the run.

-The value specified with the FILEID keyword in the PERFORM command. The value specified here overrides the Receive file name value from the mailbox (requestor) profile.

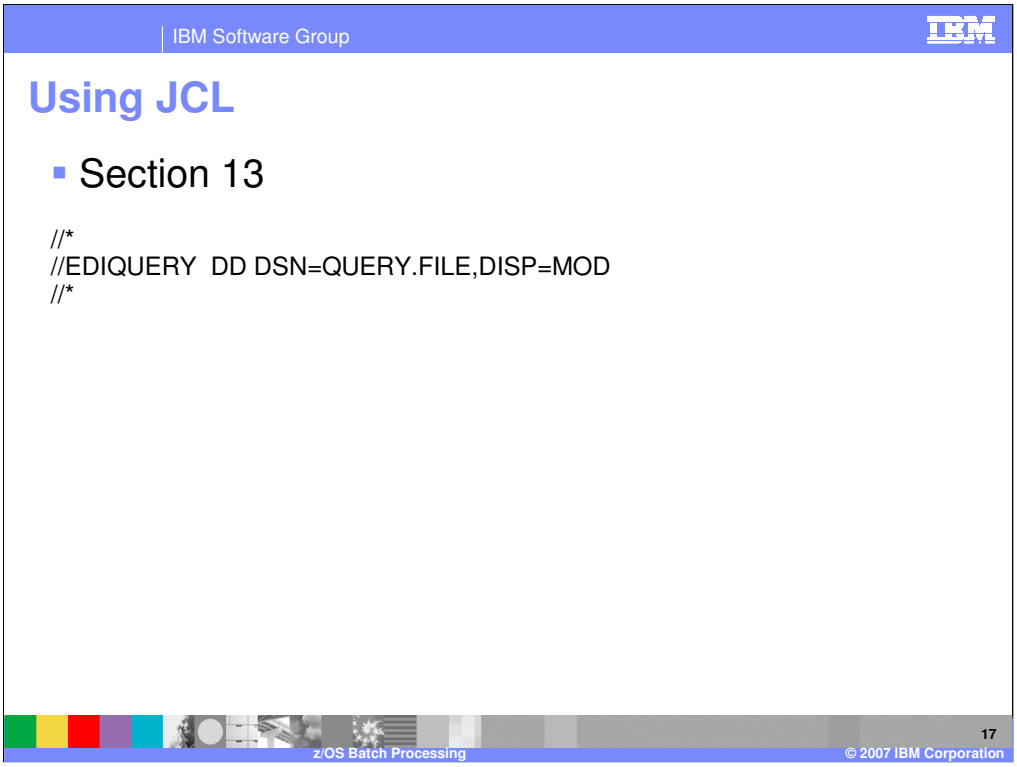
This data set can contain fixed-length or variable-length records with a logical record length of 80 or greater.



This section defines the ddname for receiving data. The ddname is specified in the Application file name field in the data format. You must specify a ddname for each data format that can be processed during the run. The name specified in the data format can be overridden with a value in the Application file name field in the receive usage/rule. If there is no applicable DD statement, the application data is written to the exception file (FFSEXCP). In the example below, INVOICE identifies a file that holds data in application format when translating from EDI standard format. This data set can contain fixed or variable length records. The logical record length must be large enough to hold the largest application record.



Report file. This section defines the file that contains reports generated by the PRINT command and by various other PERFORM commands. This data set can contain fixed block addressing (FBA) or variable block addressing (VBA) records with a logical record length of 132.



Output file. This section defines the file that contains output from command processing. EDIQUERY identifies a file that contains output from QUERY commands and from various other PERFORM commands.

Using JCL

- Section 14

```
//* If using the MAPPING MIGRATION command:                          *
//*                                                                  *
//* You may want to specify an INCNTL and/or an OUTCNTL file.        *
//* These DD names should be referenced by the INCNTL and OUTCNTL    *
//* keywords in the EDISYSIN input command language statements.      *
//*                                                                  *
//* Allocation parameters should indicate record length of at least  *
//* 80 bytes.                                                        *
//*                                                                  *
//* You should run the migration once with an OUTCNTL file           *
//* and no INCNTL file.  If changes to the mapping are needed, edit  *
//* the OUTCNTL file and submit only the changed records as the      *
//* INCNTL file the next run of mapping migration. You can use the   *
//* same file and data definition name for INCNTL and OUTCNTL.       *
//*                                                                  *
//******************************************************************
```

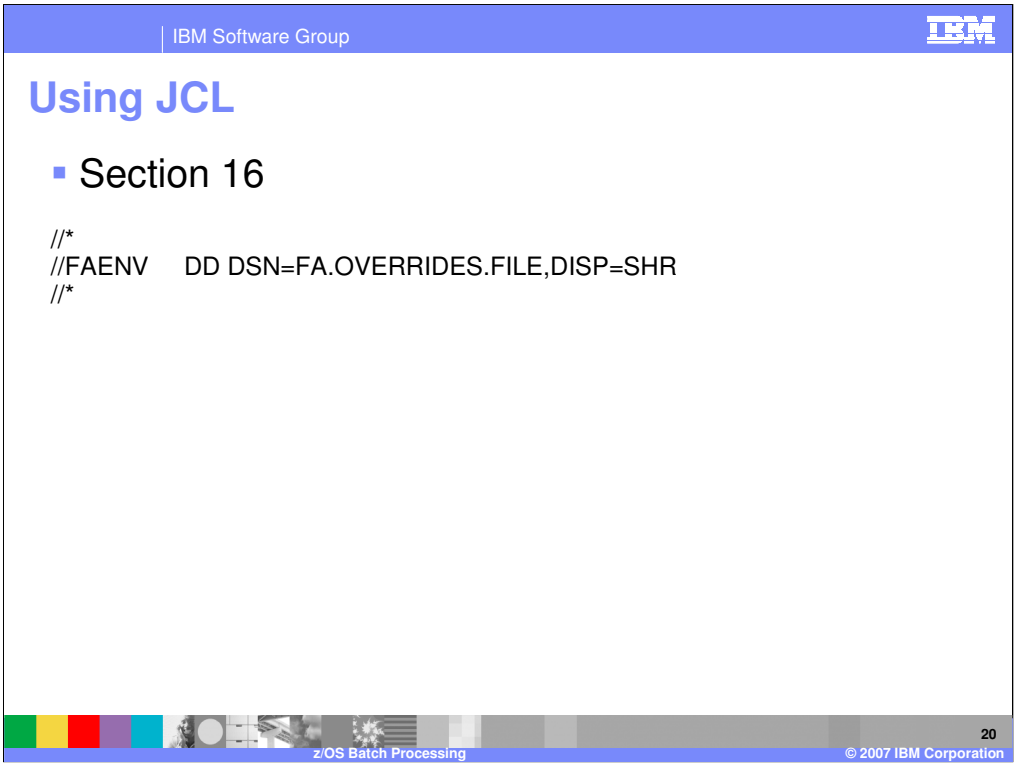

This section of JCL is required for mapping migration.

Using JCL

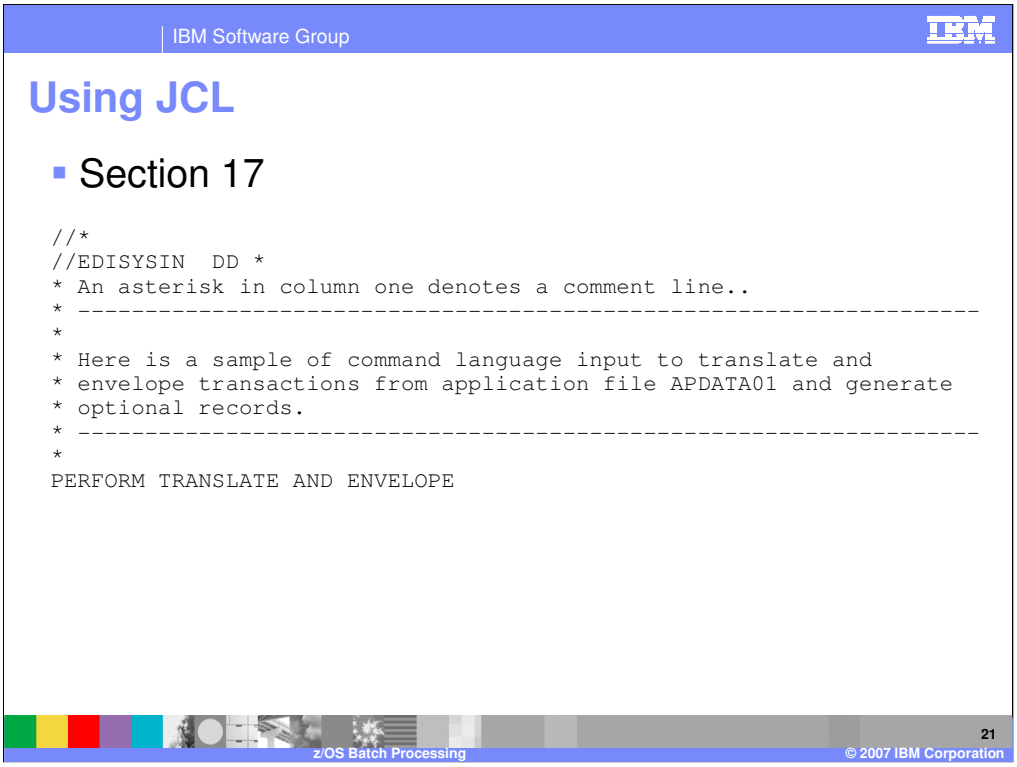
- Section 15

```
//*********************************************************************
//* If using the EXPORT or IMPORT commands:                           *
//*                                                                   *
//* 1) Allocate the export/import DI VSAM control file.               *
//*                                                                   *
//* 2) Allocate the export/import user supplied control file.         *
//*    This control file decribes what data is to be exported or      *
//*    imported. An allocation of fixed format and record length 84   *
//*    should be used. This file can also be allocated inline.        *
//*    Make sure to use the same DD name allocated with the           *
//*    CTLFILE keyword in the EDISYSIN input command language         *
//*    statements. In this sample CTLFILE(EXIMCTL) would be used.     *
//*                                                                   *
```


This section of JCL is required for exporting and importing.



This section defines override data that the translator uses to generate functional acknowledgments. FAENV is an optional file processed during deenvelope functions.



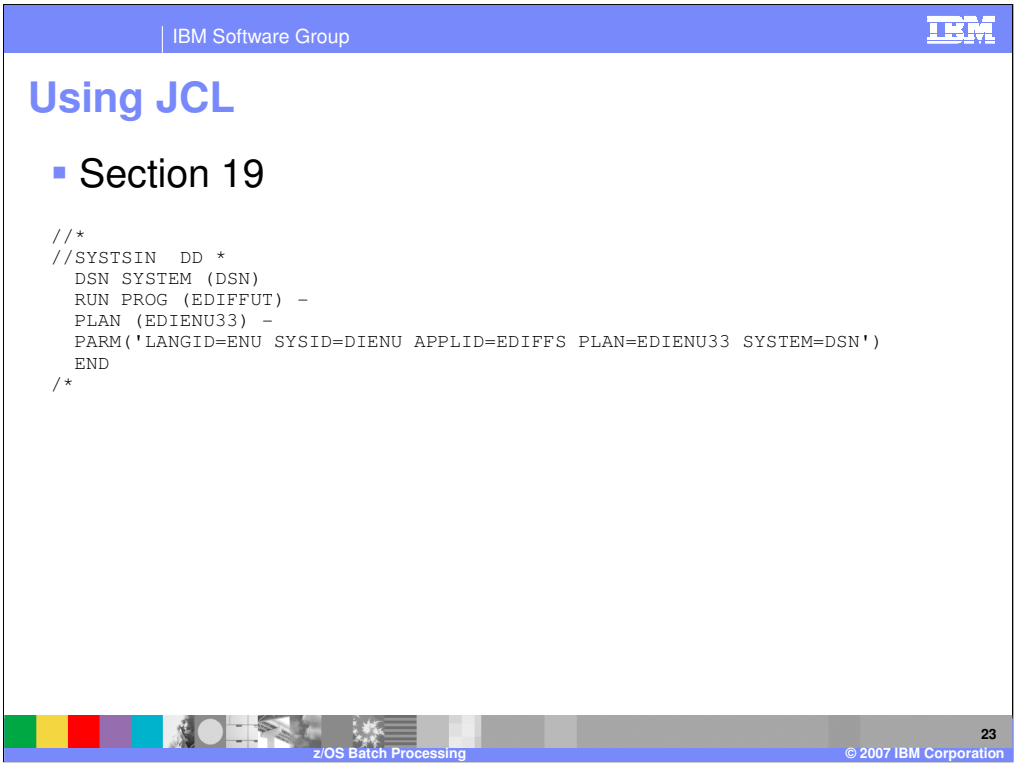
Perform command file. This section defines the file from which the Utility PERFORM commands are read. If EDISYSIN is not defined, the Utility checks SYSIN for the PERFORM commands. The command file contains the input WebSphere Data Interchange Utility commands that you want executed. The command language syntax is fairly free-form and is not case sensitive exceptions are noted in the WebSphere Data Interchange Commands and file formats reference guide. The WebSphere Data Interchange Utility first verifies that EDISYSIN is allocated. If so, the logical name EDISYSIN is used. If EDISYSIN does not exist, the logical name SYSIN is used. The command file is opened for read processing only, so it can be allocated as an inline data set in your WebSphere Data Interchange Utility JCL.

Using JCL

- Section 18

```
//*
//*EDIVAX   DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,SPACE=(CYL,25)
//*
//*********************************************************************
//* Specify the Pageable Translation work file for Data Transformation*
//*********************************************************************
//*EDIPAGE  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,SPACE=(CYL,25)
//*EDIWORK  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,SPACE=(CYL,25)
//*
//*********************************************************************
//* Specify the message parsing work file.  Uncomment the following   *
//* DD statement if you use this feature.                             *
//*                                                                   *
//* Suggested: DCB=(RECFM=V,LRECL=32760)                              *
//*********************************************************************
//*
//*EDIPARSE DD DSN=EDI.EDIPARSE.FILE,DISP=SHR
```


Work files. These files are used for pageable translation and message parsing.



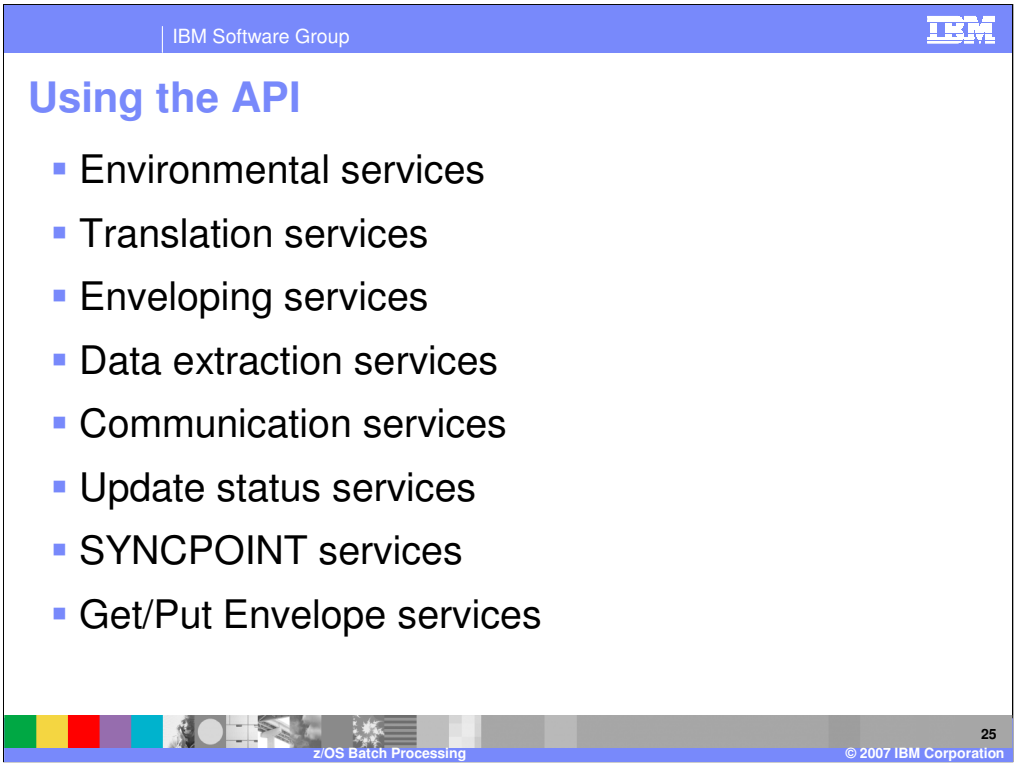
When invoking the WebSphere Data Interchange Utility, certain parameters are passed in. These parameters are keyword-oriented. The keywords should be separated by at least one blank character. This section is for DB2 only. It is typical in a DB2 environment that batch application programs (such as the WebSphere Data Interchange Utility) run under the TSO Terminal Monitor Program (TMP) in background mode. This is specified by naming IKJEFT01 in the EXEC JCL statement as in the following example:

-//XDIUTIL EXEC PGM=IKJEFT01,DYNAMNBR=50

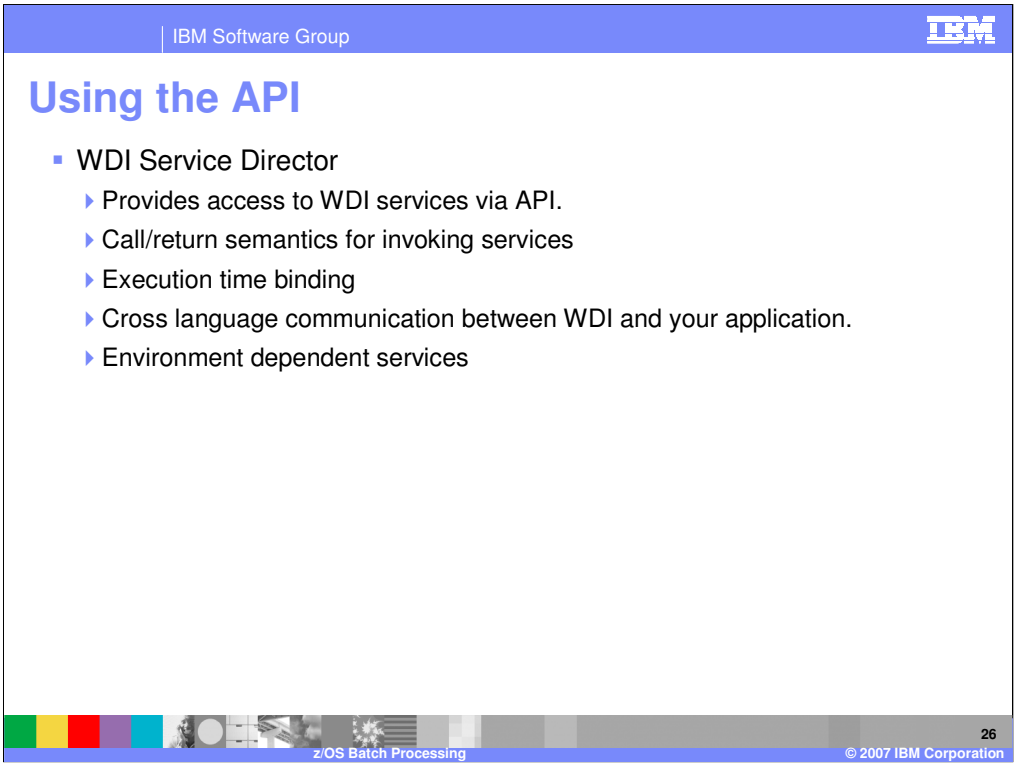
The parameters passed to IKJEFT01 are specified in the SYSTSIN data set. These parameters include the DB2 subsystem ID, the DB2 plan, the name of the application program, and the application program parameter string. IKJEFT01 connects DB2 and opens the plan, runs the application program (EDIFFUT), and then closes the plan and disconnects DB2. When EDIFFUT executes via IKJEFT01 the DB2 processing mode is called DSN. This processing mode assumes that data set EDITSIN does not exist, and that the WebSphere Data Interchange Utility parameters are specified in SYSTSIN.

Section

Using the Application Program Interface (API)



You can request WebSphere Data Interchange services directly from an application program. You can access WebSphere Data Interchange services with a simple call statement to a WebSphere Data Interchange provided *stub* program.



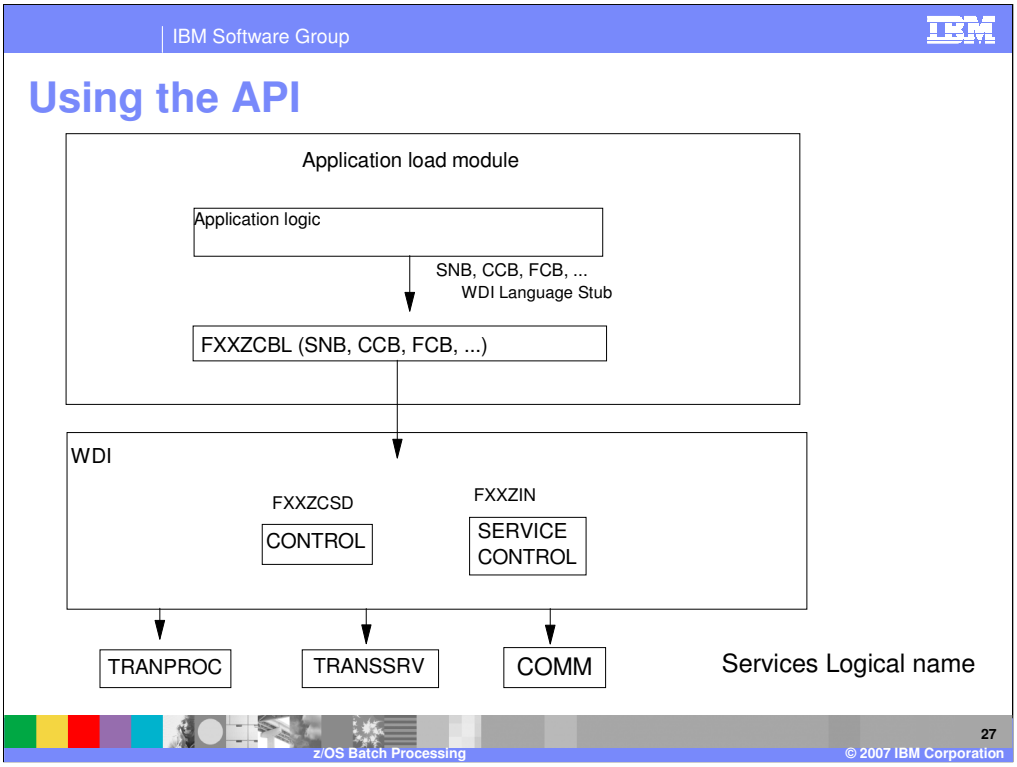
The Service Director is a set of routines which provide access to WebSphere Data Interchange services and a facility for defining those services. The functions provided by the Service Director include:

Call/return semantics for invoking services.

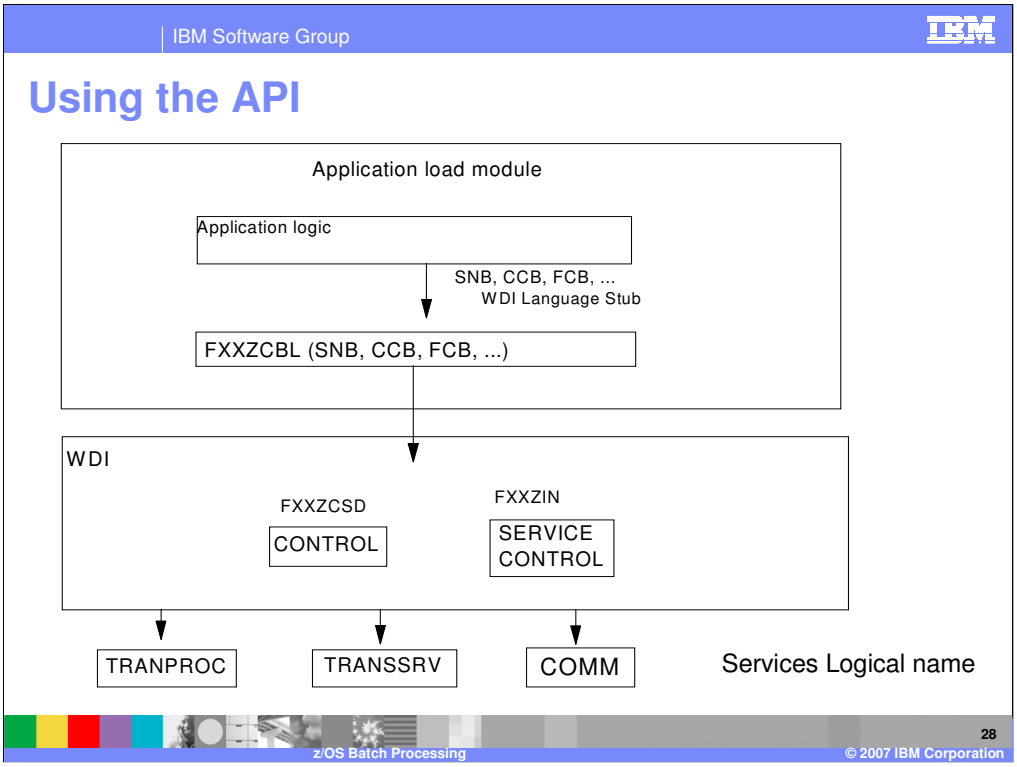
Execution time binding. The services are not linked with the calling application.

Cross language communication between WDI supported languages. (For example, a COBOL application may request a service written in "C").

Environment dependent services.

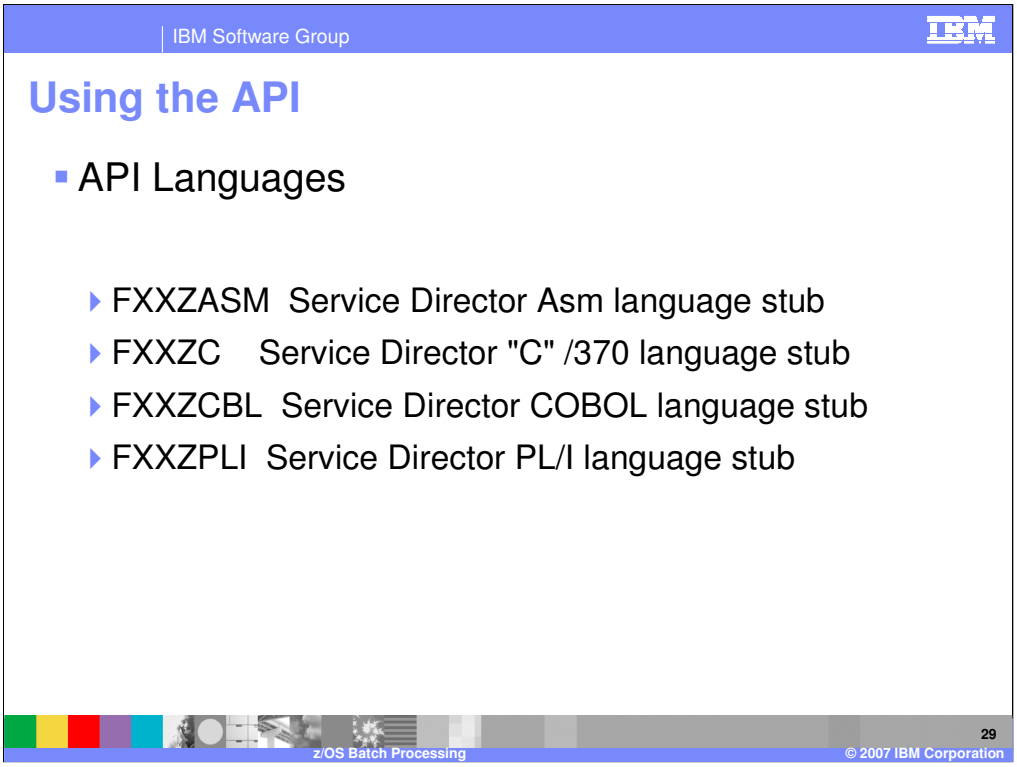


The Service Director is the "control point" and provides an access to all common services from the application. The application programmer, when requesting a service, utilizes a logical name. The logical name references the physical name of the service requested. From the application program viewpoint the services appear as "logical services". This means that the application calls the service by an assigned name with no regard to where the actual "physical" service resides or what its real physical name is. The Service Director handles all calls to the services and routes the requests on behalf of the application.



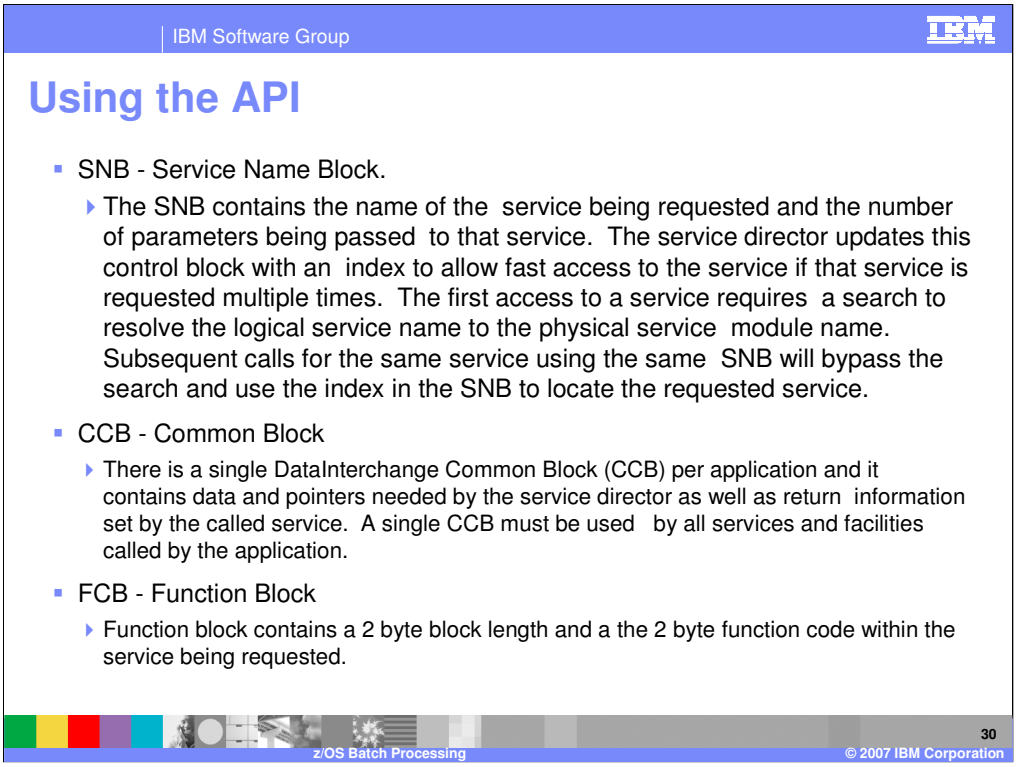
By using logical names, the application program is given the ability to perform "dynamic calls". Functions can be changed/modified/enhanced with no impact to the application. Once the service director is given control its actions are controlled by a service table. This table contains service names, locations, flags, indicators, etc.

The service director is given control by the application issuing a call to a **language stub** and passing it the required information (requested service, common block address, etc.). The service director will then map the logical to physical name, then load and pass control to the service. When the service has completed processing, control returns to the service director and then to the application program. The application can then examine the return code, data buffers, etc., and continue processing.



The load library distributed with WDI contains a load module for each of the stub programs. These programs are linked with the application program requesting the service. These stub programs are pieces of code that load and transfer control to WDI when a request is made. WDI is not physically part of the application load module. The distributed load library should be part of the //SYSLIB for the linkage editor.

Although WebSphere Data Interchange directly supports only these languages, any language that can create an operating-system-style parameter list, and that uses operating system linkage and register conventions, can request WebSphere Data Interchange services by using the FXXZASM stub program.



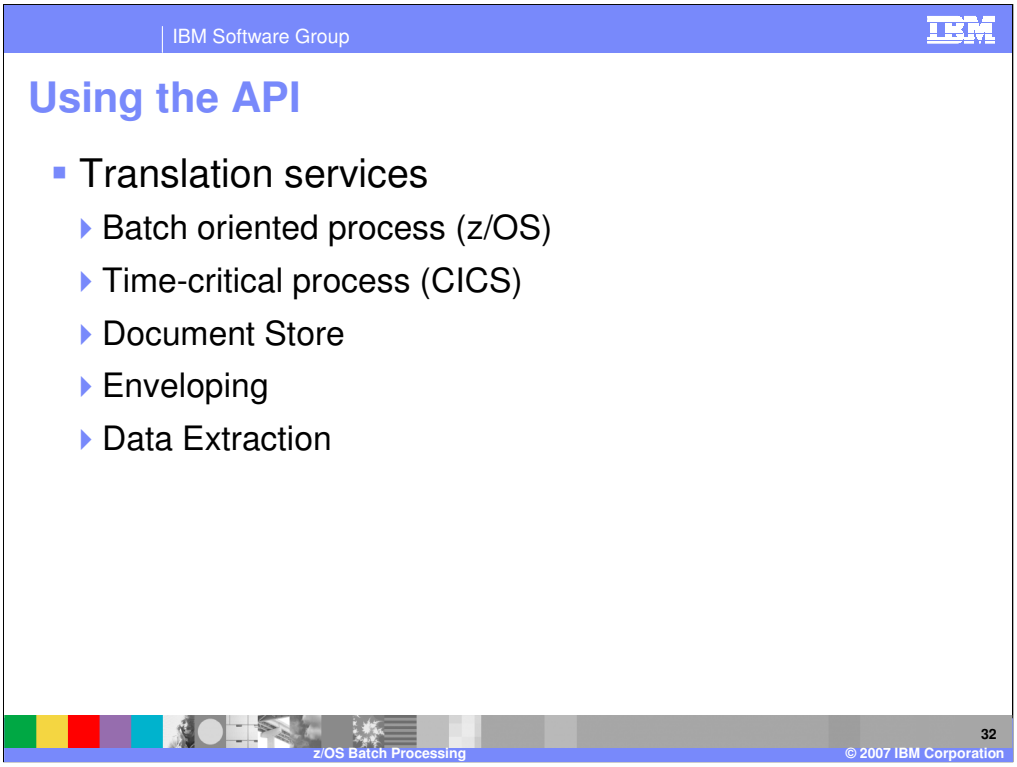
WebSphere Data Interchange assumes that all parameters are pointers to control blocks. Any remaining parameters have no meaning to the service director but are parameters that will simply be forwarded to the requested service.

Using the API

- Environmental services
  - Initialization
    - FXXZccc (SNB, CCB, FCB, 'applid', 'sysid')
  - Termination
    - FXXZccc (SNB, CCB, FCB)
  - Utility Service
    - FXXZ*ccc* (SNB, CCB, FCB, UTILCB)

Environmental services both establishes and removes the WebSphere Data Interchange environment. First, your application program must request initialization. When initialization is complete, you can request other services. If you request another service before requesting an initialization, either a return code of -1 is posted in the CCB, or the program ABENDs. Finally, your application program must request termination. The services requested between initialization and termination (such as translation or communication services) internally acquire storage, open files, and obtain control over resources. Requesting termination is necessary so WebSphere Data Interchange can release all resources that it has obtained.

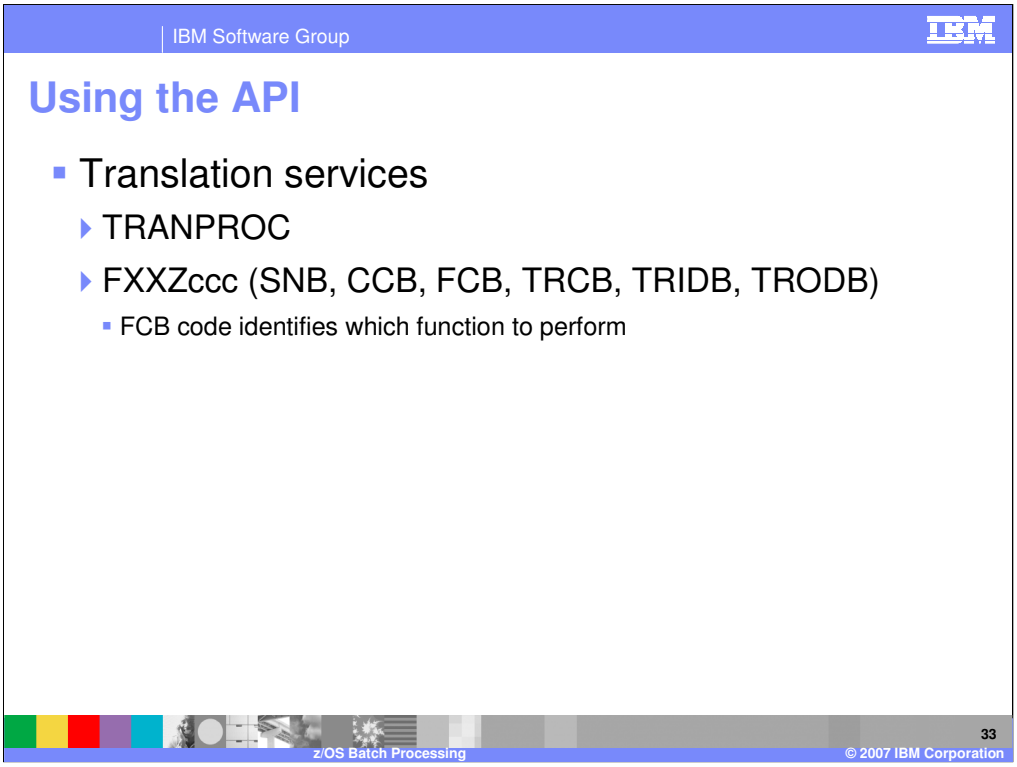
The Utility Service API is a special service and is used as a performance technique available with WDI in CICS. This technique is identified as “Hot DI”. “Hot DI” is a term used with WebSphere Data Interchange to describe a CICS environment whereby WDI processing “threads” are always initialized. Initialization occurs when WDI is getting starting. The product obtains control storage, determines some “instance options”, and prepares to process data. In a CICS environment there normally are many small transactions. By reducing the number of times WDI has to initialize itself, the time to process transactions after the first one can be significantly reduced. In this case WDI is always ready to process and already “warmed-up” \; hence the designation “Hot DI”.



Translation services, enveloping services, and data extraction services are closely related. They share an API implemented by the same logical service (TRANPROC). These services also share the Document Store. Exchanging information with a trading partner is a sequence of interruptible events whose status must be maintained and tracked. In the batch-oriented process, application data is stored in files owned by the applications until the next batch job runs, which translates the application data into an EDI standard format. This job, or a different job run later, can request that all the EDI standard data be gathered (enveloped) and sent to the trading partner. The trading partners receive the data, process the data, and then return a response.

In the time-critical process, application data is usually translated, enveloped, and sent as soon as the data is available. Although some batching of data might occur, the batches are typically much smaller than in the batch-oriented process. The trading partner is expected to process the data and generate a response immediately.

The Document Store database saves all EDI standard data sent or received and tracks the status of the data as it progresses from translated, to send, to received, to acknowledged. The Document Store is updated by all translation and enveloping operations, and is maintained and reported on by the Document Store functional area or the WebSphere Data Interchange Utility. Updating the Document Store database is optional and can be controlled using the Application Defaults (APPDEFS) profile. For more information about the Application Defaults profile, refer to the WebSphere Data Interchange User's Guide.



The logical name for the translation service is TRANPROC. Some examples of codes and functions (FCB function code) provided by the translation service are:

-**111** Test translate-to-EDI-standard

-**131** Production translate-to-EDI-standard

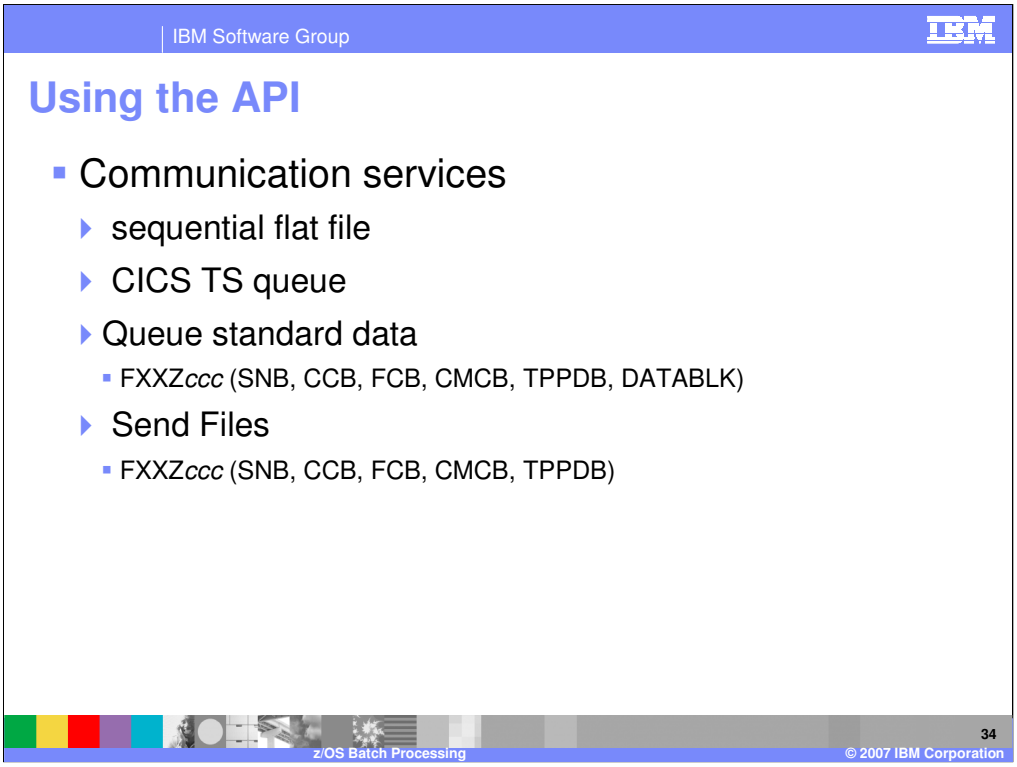
-**211** Test deenvelope and translate-to-application

-**212** Production deenvelope and translate-to-application

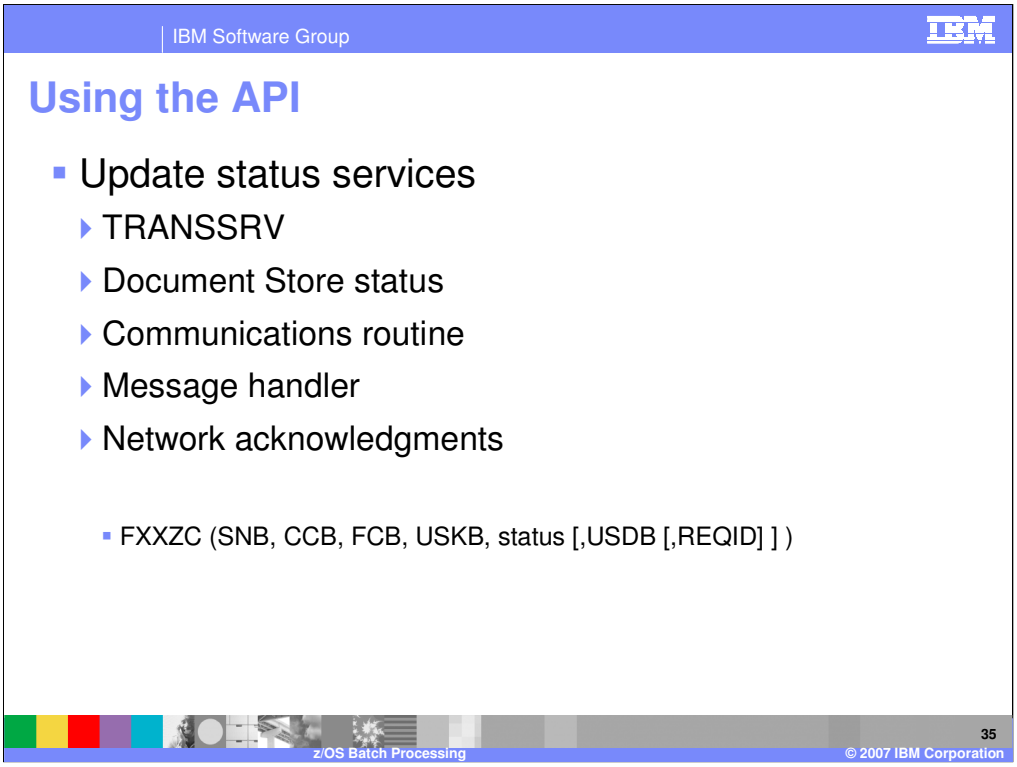
-**215** Envelope data

-**216** Detailed transaction data

-**1000** End translation

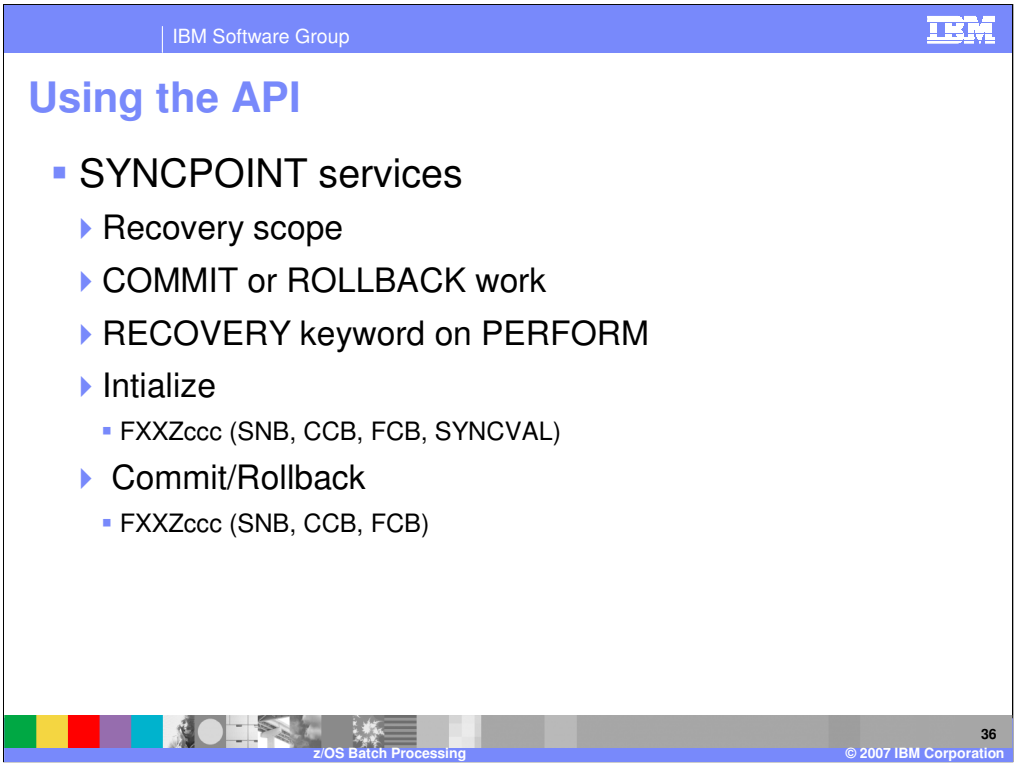


The functions provided in the Communication API enable an application program to send or receive transaction data, files, and messages to and from trading partners. The transaction or file data that is to transmitted is not passed directly through an API parameter. Rather, it is provided indirectly as the name of a sequential flat file that contains the data to be sent or must be used to collect the data that is being received. In CICS, the data to be sent or received is restricted to a TS queue.

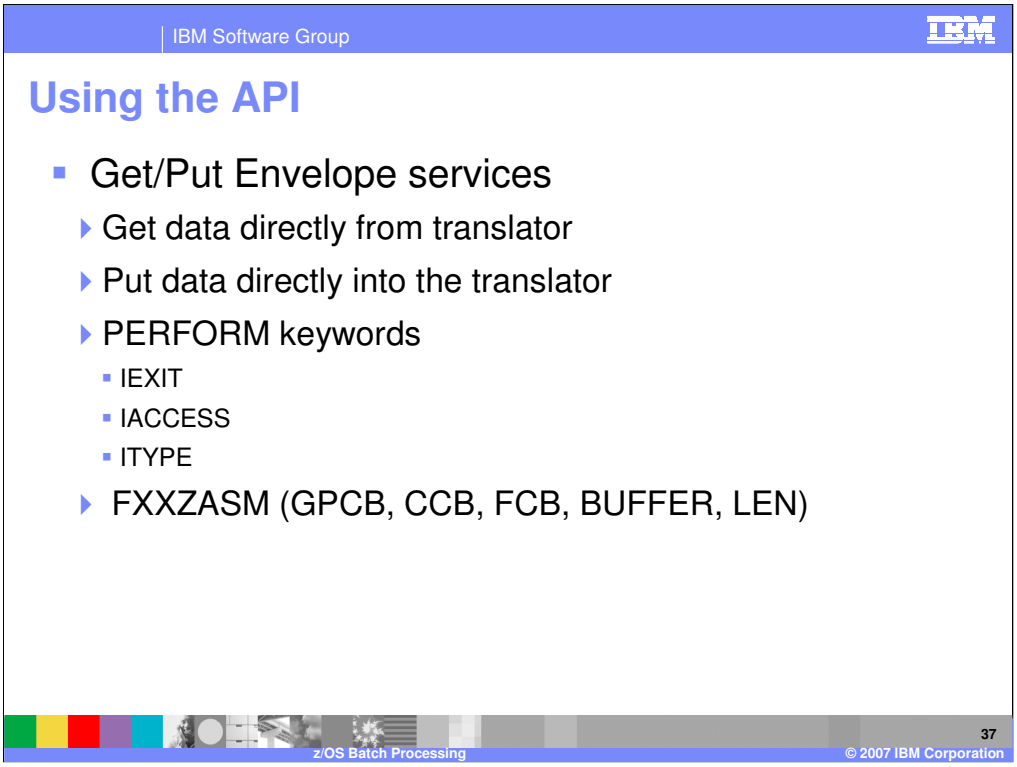


Update status services allow the status of an interchange to be updated as it progresses from enveloping, to sending, to receipt by the trading partner. The update status service is used by the communications routine and message handler when an interchange is sent, and also during the processing of network acknowledgments. Certain characteristics of an interchange become known only when the interchange is sent. Therefore, the update status service also allows other fields in the interchange to be set when status is updated. These are established using the update status data block. The logical name for the update status service is TRANSSRV.

The FCB function code identifies the type of key provided. The USKB is the key block with a format based on the value in the FCB. Using the update status data block (USDB) is always optional. This data block contains information about an interchange that is generally determined at the time an interchange is sent. With REQID, the management reporting component of WebSphere Data Interchange is called to update the statistics on the number of bytes sent from the specified requestor ID.



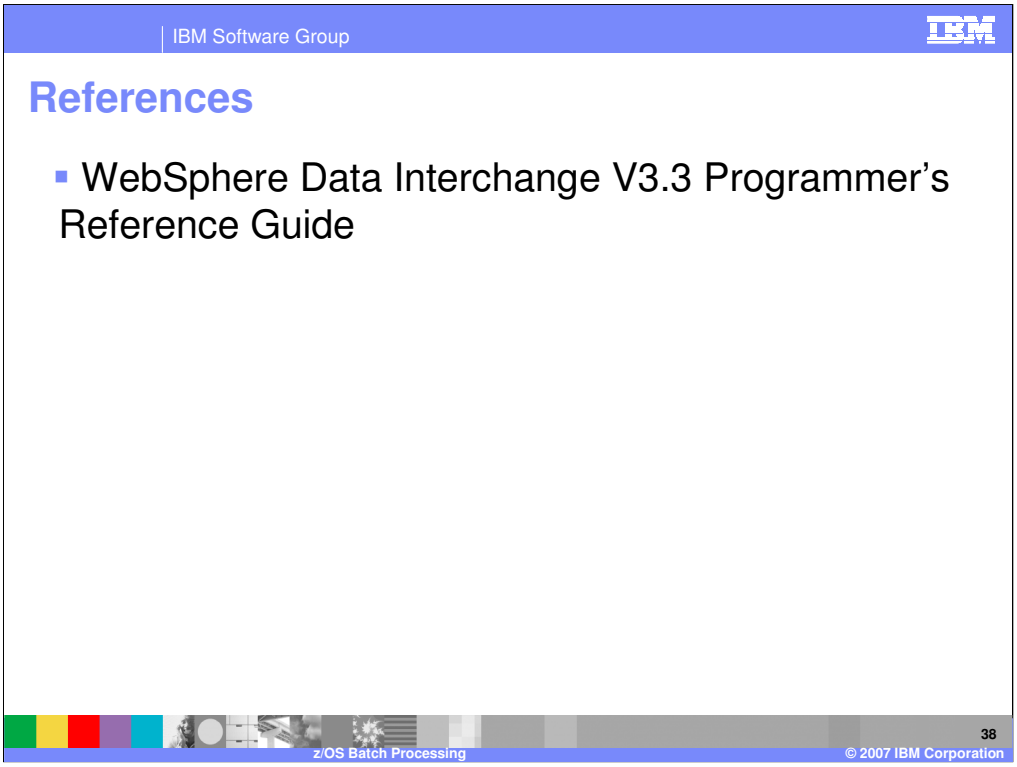
SYNCPOINT services provide the interface for controlling the recovery scope, and provide an environment-independent interface for requesting either that changes to resources are permanent (COMMIT work) or that changes must be removed (ROLLBACK work). If a system or application fails, or if a specific ROLLBACK work request is made, all changes made to resources since the last COMMIT point are backed out of the system. It will be as if the changes were never made. In an z/OS/DB2 environment, a request to commit work is made directly to DB2 using a COMMIT request. A request to roll back work is made directly to DB2 with a ROLLBACK request. In a CICS/DB2 environment, a request to commit work is made to CICS with an EXEC CICS SYNCPOINT request. CICS passes this request on to DB2 and controls the committing of CICS resources with DB2 resources. A request to roll back work is made to CICS with an EXEC CICS SYNCPOINT ROLLBACK request. Again, CICS passes this request to DB2 and controls the removal of CICS resources with DB2 resources. SYNCPOINT services issues the proper request based on the execution environment. When you use this API, recovery scope is established using the SCOPE field in the TRCB. When you use the WebSphere Data Interchange Utility, recovery scope is established by using the RECOVERY keyword on the PERFORM command.



During outbound processing, you can use the Get envelope service to get data directly from the translator, rather than from the file the translator writes to. Normally, after WebSphere Data Interchange creates an interchange, the interchange is routed automatically to a file. (In CICS, this file is always a TS queue.) By writing an exit to circumvent this action, the envelope can be retrieved directly from the translator and then processed. In CICS, for example, the interchange could be routed to a TD queue.

During inbound processing, you can use the Put envelope service to put data directly into the translator, rather than having the translator read a file. Normally, in order to deenvelope an interchange, WebSphere Data Interchange reads a file containing the interchange. In CICS, this file is always a TS queue. By writing an exit to circumvent this action, you can pass an envelope directly into the translator. In CICS, for example, an interchange could be read from a TD queue and passed directly into the translator.

WebSphere Data Interchange recognizes this type of user exit when the following keywords are used on enveloping commands: IEXIT(*exitname*), IACCESS(**M**), and ITYPE(**UE**).



More information can be found in the WebSphere Data Interchange Version 3.3 Programmer's Reference Guide.

Template Revision: 04/25/2006 11:09 AM

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	WebSphere MQ	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and 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 and/or changes in the product(s) and/or program(s) 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.

Information is provided "AS IS" without warranty of any kind. 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 (e.g., 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.