POWERSERVER

(Version 7.63)

This page intentionally left blank.

TABLE OF CONTENTS

Description **Page**

TABLE OF CONTENTS (continued)

Description **Page**

CHAPTER 1 INTRODUCTION

This chapter will cover the general concepts of the **POWERServer** module. This document describes a command protocol for performing operations on a Data Pro database accessed via **TCP/IP**. The client program submits a request to the server consisting of a command to be executed. The server provides a response to be interpreted by the client. The only pre-requisite required to use the **POWERServer** module is that the **Advanced Security Administrator** module must be installed and running. A specific **User ID** must be assigned to handle the **TCP/IP** calls from the client to the **POWERServer**.

POWERSERVER CONFIGURATION

There are four data items that need to be set up for the **POWERServer** Configuration.

LOGIN USER NAME	The **Login User Name** is the **User ID** set up in the **Advanced Security Administrator** module for the specific user to be called to communicate between the client and the **POWERServer**.
LOGIN USER PASSWORD	The **Login User Password** is the password that was assigned to the specific **User ID** in the Advanced Security Administrator that corresponds with the **Login User Name**.
IP SOCKET PORT	The **IP Socket Port** is the port number that is defined to use to communicate through the TCP/IP protocol. This port number should match the port number, used by the client application when initializing the TCP/IP connection to the **POWERServer**.
ACTIVITY LOGGING	The **POWERServer** module can create a log file, which contains information about logins and activity *(requests and responses made to and from the **POWERServer**)*. There are three choices for logging the **POWERServer** activity. **Disable Log** **Log Connections Only** **Log All Activity** **NOTE:** When you choose to log **"All Activity"** the **POWERServer** log file can become very large in size. It is suggested that this should only be done during a testing or debugging phase of the client application.

START POWERSERVER PROCESS

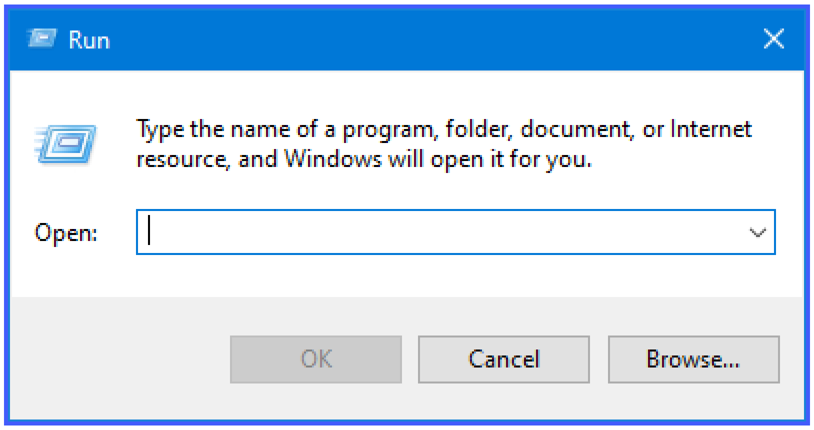
Once the configuration section is completed and saved you may then start the **POWERServer** process. The system will display a box on your screen and ask **"Do you wish to start it now?"** Click on the **"Yes"** button and a new application window will appear on your desktop. This window must be kept open as it is waiting to receive requests from the client.

SETUP POWERSERVER AS A WIN32 SERVICE ON THE SERVER

One of the servers or workstations on your network must now be chosen as the **POWERServer "server."** This is the server that runs the **POWERServer** process in the background, listening for connection requests from the client application and then accessing the accounting system. It is recommended that you choose the same network server that the accounting system is installed on.

To Setup POWERServer:

1. From the Accounting Server, <click> the **Start** button and select the **Run** menu option from the "Windows Systems" menu options. The **"Run"** window displays.

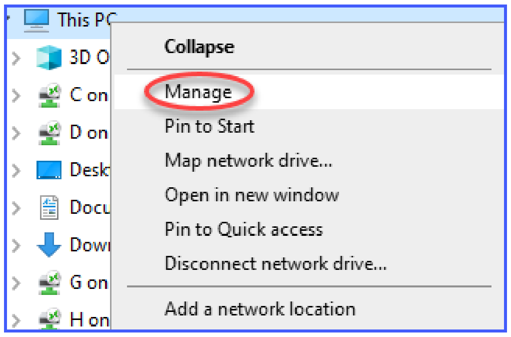

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
OK
Cancel
Browse...

2. Type "**cmd**" in the **Open** text box.
3. <Click> the **OK** button. The Command Prompt window displays.
4. Change the directory to where the accounting system is installed ***(i.e. cd c:\apps\power)***. The prompt changes to the accounting software directory.
5. From the command prompt in the accounting software directory, type **"POWERService install"** and press **Enter** on the keyboard. This command installs the **POWERServer** module as a Windows Service which automatically starts whenever your server is turned on or rebooted.
6. Type "**exit**" and press **Enter** on the keyboard to close the command prompt window.

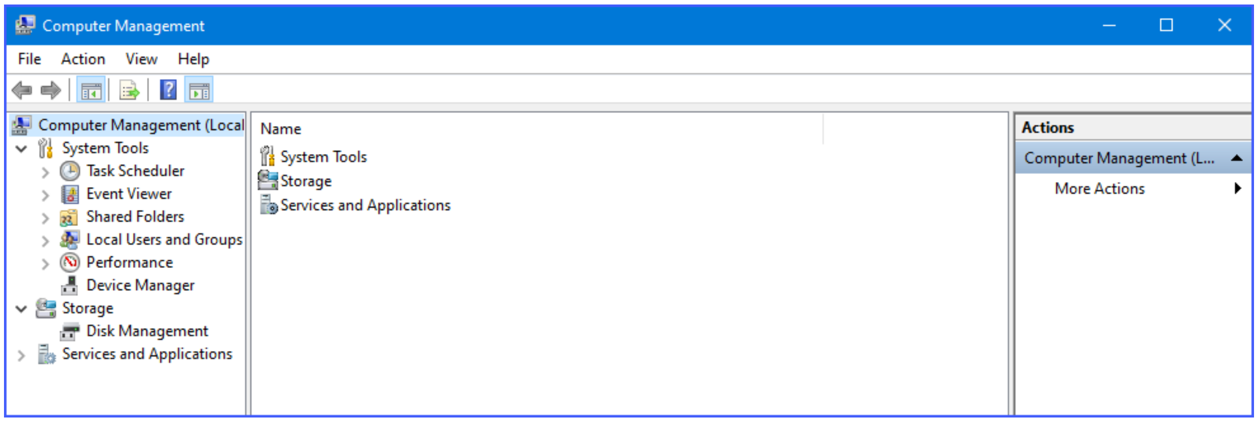
To Begin POWERServer for First Time:

Once the **POWERServer** has been installed as a service, you must start the process for the first time.

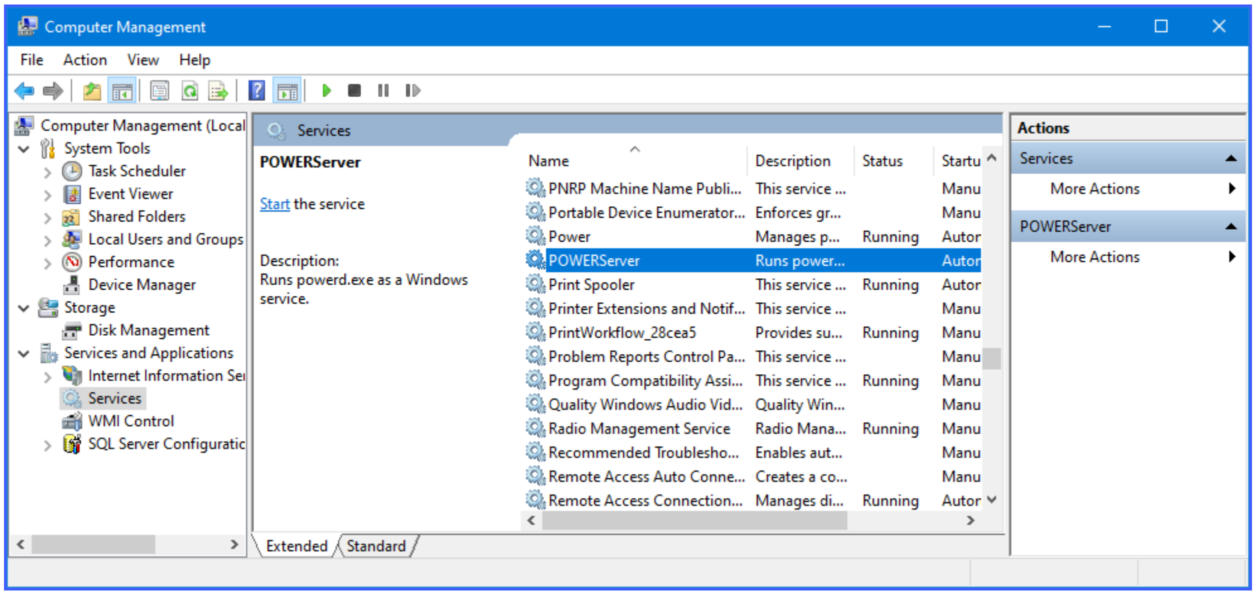
1. From your Windows desktop, <right-click> on the **"This PC"** option and select the **Manage** menu option.



The **Computer Management** window displays.



2. From the menu on the left, <click> the **Services and Applications** option and then select the **Services** sub-option. A list of services installed on your system displays in the right window.
3. Scroll down to the **POWERServer** service, <right-click> and then select the **Start** menu option. The **POWERServer** utility starts and returns to the Computer Management window.



Description:
Runs powerd.exe as a Windows service.

Name	Description	Status	Startu
PNRP Machine Name Publi...	This service ...		Manu
Portable Device Enumerator...	Enforces gr...		Manu
Power	Manages p...	Running	Autor
POWERServer	Runs power...		Autor
Print Spooler	This service ...	Running	Autor
Printer Extensions and Notif...	This service ...		Manu
PrintWorkflow_28cea5	Provides su...	Running	Manu
Problem Reports Control Pa...	This service ...		Manu
Program Compatibility Assi...	This service ...	Running	Manu
Quality Windows Audio Vid...	Quality Win...		Manu
Radio Management Service	Radio Mana...	Running	Manu
Recommended Troublesho...	Enables aut...		Manu
Remote Access Auto Conne...	Creates a co...		Manu
Remote Access Connection...	Manages di...	Running	Autor

PRINT POWERSERVER LOG FILE

You may print the **POWERServer** log file at any time by choosing this option. This option will also allow you to remove the **POWERServer** log file. When you remove the log file, it will be re-created upon the next call to it from the client. Please keep in mind that if you are logging **"All Activity,"** this file can become very large in size.

CHAPTER 2 COMMAND LANGUAGE

This document describes a command protocol for performing operations on a Data Pro database accessed via TCP/IP. The client program submits a request to the server consisting of a command to be executed. The server provides a response to be interpreted by the client.

A server request is a string consisting of a request type and optional request parameters. The request type and request parameters are delimited by a tilde **(~)**. Request parameters consist of one or more comma-delimited keywords.

If a keyword contains a series of comma-delimited values, the series must be enclosed in parentheses. Embedded spaces are significant. Fields that might contain a comma should be enclosed in graves **(`)**. Requests and request keywords are case-insensitive.

A server response is a string consisting of a result code and additional information fields appropriate to the response. If the request succeeds, the result code is **0**. Otherwise, the result code is non-zero. The result code and each of any subsequent fields are tilde-delimited. Additional information fields provided in a successful response are described in the section for the particular request type. Result codes representing errors are described in the Error Codes Section at the end of this document.

The syntax and functionality of the available request types are described on the following pages.

ADDREC

Syntax:

ADDREC~FILETYPE=xxxx,FIELDS=(...),DATA=(...),

[,LOCKREC=(FILETYPE=xxxx[,KEYNO=n][,KEY=(...)])],OP=x

This request adds a record of the specified file type to the data base. The record consists of one or more data values corresponding to the specified data base field names.

A single **ADDREC** transaction may be submitted as a single request or as a series of requests to the server. The server recognizes completion of the transaction when it receives the **OP** *(operation)* keyword with the key value **COMMIT**. At that time, the server processes the entire **ADDREC** transaction and performs the requested operation.

If a single **ADDREC** transaction is submitted as a series of requests, the calling program must observe a number of rules. Each keyword must be contained entirely within the request *(i.e. the keyword cannot be broken across requests)*. With the exception of the **OP** keyword (which is present on the final submission), keywords may be specified in any order. The **DATA** and **FIELD** keywords may be specified multiple times within the transaction.

If this is the case, the server treats the multiple occurrences as a single occurrence by concatenating them in the order in which they were submitted. The caller should check the server response after each submission for a successful result code. Because the server does not actually process the transaction until receiving a **COMMIT** operation, the final response string contains explanatory information if there is an error.

The following keywords may be used:

FILETYPE	A required keyword designating the file to which the record should be added.
FIELDS	A required keyword consisting of one or more comma-delimited data base field names to be included in the record. The corresponding **DATA** keyword must contain a value for each field name specified.
DATA	A required keyword consisting of one or more comma-delimited data values to be included in the record. The corresponding **FIELDS** keyword must contain a field name for each data value specified.
OP	A keyword that is specified at the time the server should actually process the transaction. May have the value **COMMIT** (process the transaction) or **ABORT** (cancel the transaction).
LOCKREC	An optional keyword that specifies an associated record to be locked while the **ADDREC** takes place. **LOCKREC** may contain the following keywords:
FILETYPE	A required keyword designating the file type of the associated record to be locked.
KEYNO	An optional keyword for indexed files designating the key number to be used to identify the associated record to be locked.
KEY	A required keyword for indexed files identifying the associated record to be locked.

When the caller commits the transaction, the server performs syntax checking and data validation. The following validation tests are performed:

- The record to be added must not already exist.
- Each database field name must exist in the data base.
- The data types of provided data values must match the type of their associated data dictionary definition *(i.e., an alphanumeric string cannot be provided for a numeric value)*.
- For indexed files, data values must be provided for each primary key.
- Additional file and field-specific **"reasonableness"** tests.

If no syntax or validation problems are encountered, the record is added. Otherwise, the transaction is rejected.

Example:

Add a record to file GL01. The account number (primary key) is 150:

```
ADDREC~FILETYPE=GL01,FIELDS=(account,desc),DATA=(150,Cash),OP=COMMIT
```

Add an SO02 detail record. Lock the SO01 record during the add:

```
ADDREC~FILETYPE=SO02,LOCKREC=(FILETYPE=SO01,KEY=10001)
ADDREC~FIELDS=(sonum,linenum,holdyn,sc,desc)
ADDREC~DATA=(10001,6,n,10000,`deluxe widgets`)
ADDREC~FIELDS=(taxyn,qtyord,weight,disc,price)
ADDREC~DATA=(y,2,5.5,5.0,14.95)
ADDREC~OP=COMMIT
```

CHECKRIGHTS

Syntax:

CHECKRIGHTS~SYSTEM=x,OPTION=x

This request allows the client to determine if the currently logged-in user has rights to access the specified system and option.

The following keywords may be used:

SYSTEM	A required keyword designating the application system.
OPTION	A required keyword designating an option ID within the specified system.

Example:

```
CHECKRIGHTS~SYSTEM=GL, OPTION=GL0101
```

Determines whether the currently logged-in user has access rights to the General Ledger

Option GL0101. If the user has rights, the server response string is 0.

See Also:

CRFILEDEF, DELFILEDEF, DELFILE

CRFILE

Syntax:

CRFILE~FILETYPE=x

This request creates a data file based on the specified custom file definition. If the file pattern contains tokenized elements, the ***SET*** command should be used prior to invoking ***CRFILE*** to initialize the appropriate system settings. ***CRFILE*** must be invoked before attempting to use any of the other data manipulation commands on a custom file. The following keyword may be used:

FILETYPE	A required keyword representing the file type of the file to be created.

Example:

SET~COMPANY=abc
CRFILE~FILETYPE=EMP Create the data file associated with the file type EMP.

If file pattern associated with *EMP* is *coEMP*, the file that is created is named *abcEMP.* Should be taxed.

See Also:

CRFILEDEF, DELFILEDEF, DELFILE

CRFILEDEF

Syntax:

CRFILEDEF~FILETYPE=x,

[DESC=x,]

PATTERN=x,

FIELD=(fieldname,description,datatype,arraysize1,arraysize2,length,nbrdecimals),

[INDEX=(indexno, uniqueYN,description),]…

[SUBINDEX=(indexno,subindexno, fieldname,description,capitalsYN),]…

OP=x

This request defines the structure of a custom file and saves the definition for later use. The ***CRFILE*** request must subsequently be invoked to create the file. Then ***ADDREC**, **GETREC**, **DELREC**,* and ***UPDREC*** requests may be used to maintain data in the file.

A single ***CRFILEDEF*** transaction may be submitted as a single request or as a series of requests to the server. In either case, keywords may be specified in any order.

If a single ***CRFILEDEF*** transaction is submitted as a series of requests, the calling program must observe a number of rules. Each keyword must be contained entirely within the request *(i.e., the keyword cannot be broken across requests)*. With the exception of the ***OP*** keyword *(which is present on the final submission)*, keywords may be specified in any order. The ***FIELD***, ***INDEX**,* and ***SUBINDEX*** keywords may be specified multiple times within the transaction. The caller should check the server response after each keyword submission for a successful result code.

After the transaction is committed, the server performs complete transaction checking and executes the request.

The following keywords may be used:

FILETYPE	A required 1-4 character keyword identifying the file type. The file type must not already be in use by the database, either as a user-defined or Data Pro built-in file type.
DESC	An optional 1-30 character keyword describing the file type.
PATTERN	A required 1-8 character keyword specifying the name of the file that will contain data for the file type. The pattern may contain standard Data Pro token elements *(for example,* ***"co", "dy",*** *etc)* which will be translated to the values in use by the system at the time the ***CRFILE*** for the file type is issued. The pattern cannot match a standard Data Pro pattern.
FIELD	This keyword contains multiple values describing a field in the file. At least one field must be entered. Up to **200** fields may be specified. If a field value is omitted, the value must still be delimited by a comma. A field definition is composed of the following entries:
FIELDNAME	A required 1-10 character name unique within the file type. If the field is a single dimension array, the maximum length is **8** characters. If the field is a double dimension array, the maximum length is **6** characters.
DESCRIPTION	An optional 1-30 character field description.
DATATYPE	The field's data type. Must be one of the following: **CCHR** (character), **CSTR** (string), **CFLT** (float), **CDBL** (double), **DPDT** (date), **DPYN** (Y/N), **CINT** (small integer), **CLNG** (long integer).
ARRAYSIZE1	A required integer >= 1. If > 1, the field is a double dimension array.
ARRAYSIZE2	A required integer >= 1. If arraysize1 is 1, but arraysize2 > 1, the field is a single dimension array.
LENGTH	An integer >= 1. Ignored for data types **DPDT** and **DPYN**. Required for other data types.
NBRDECIMALS	An integer >= 0. Required for data types **CFLT** and **CDBL**. Ignored for other data types.
INDEX	An optional keyword containing multiple values describing an index on the file. Up to 5 indexes may be entered. If an index value is omitted, the value must still be delimited by a comma. If an index is defined, at least one **SUBINDEX** keyword must be defined for that index. An index definition is composed of the following entries:
INDEXNO	A required integer from 1 to 5. Index numbers must be assigned consecutively beginning with 1.
DESCRIPTION	An optional 1-30 character index description.
UNIQUEYN	A required entry, either **'Y'** or **'N,'** designating whether the key values in the index are unique.
SUBINDEX	A multiple-value keyword that is required if the keyword ***INDEX*** is defined. The keyword describes a sub-key that makes up an index. Up to 5 **S*UBINDEX*** keywords may be specified for a single index. If a sub-index value is omitted, the value must still be delimited by a comma. A sub-index definition is composed of the following entries:
INDEXNO	A required integer from 1 to 5. Must refer to the index number of the associated **INDEX** keyword.

SUBINDEXNO	A required integer from 1 to 5 representing the order of the sub-index within the index. Sub-index numbers must be assigned consecutively beginning with 1.
FIELDNAME	The database field to be used as a basis for the sub-index. The field name must be defined using a **FIELD** keyword. If the field is defined as an array, the standard Data Pro array notation must be used in specifying the field name *(e.g. fieldname[n]).*
DESCRIPTION	An optional 1-30 character sub-index description.
CAPITALSYN	A required entry, either **'Y'** or **'N,'** designating whether the data in the sub-index should be translated to upper-case.
OP	A keyword that is specified at the time the server should actually process the transaction. May have the value **COMMIT** *(process the transaction)* or **ABORT** *(cancel the transaction).*

Example:

CRFILEDEF~PATTERN=coEMPdy,DESC=Employee List

CRFILEDEF~INDEX=(1,Employee ID,Y)

CRFILEDEF~SUBINDEX=(1,1,id,Id,Y)

CRFILEDEF~FIELD=(id,Employee ID,CSTR,1,1,10)

CRFILEDEF~FIELD=(wage,Employee Hourly Wage,CDBL,1,1,15,2)

CRFILEDEF~FILETYPE=EMP,OP=COMMIT

SET~COMPANY=abc,DATE=010102

CRFILE~FILETYPE=EMP

This series of requests defines a file type EMP and then creates the file abcEMP02 based on the file type definition and current system settings. The file is indexed on the *id* field.

See Also:

DELFILEDEF, CRFILE, DELFILE

DELDATA FILE

Syntax:

DELDATAFILE~FILENAME=x

This request deletes the specified data file from the server directories.

The following keyword may be used:

FILENAME	A required keyword representing the name of the file to be deleted.

Example:

DELDATAFILE~FILENAME=rwreport.dat

Deletes the data file *rwreport.dat* from the server directories.

See Also:

GETDATAFILE

DELFILE

Syntax:

DELFILE~FILETYPE=x

This request deletes the data file associated with the specified custom file definition. If the file pattern contains tokenized elements, the ***SET*** command should be used prior to invoking ***DELFILE*** to initialize the appropriate system settings.

The following keyword may be used:

FILETYPE	A required keyword representing the file type of the file to be deleted.

Example:

```
SET~COMPANY=abc
DELFILE~FILETYPE=EMP
```

Delete the data file associated with the file type *EMP*. If the file pattern associated with *EMP* is *coEMP*, the file that is deleted is named *abcEMP*.

See Also:

CRFILEDEF, DELFILEDEF, CRFILE

DELFILEDEF

Syntax:

DELFILEDEF~FILETYPE=x

This request deletes the specified custom file definition. The following keyword may be used:

FILETYPE	A required keyword representing the file definition to be deleted.

Example:

```
DELFILEDEF~FILETYPE=EMP
```

Deletes the file definition *EMP*.

See Also:

CRFILEDEF, CRFILE, DELFILE

DELREC

Syntax:

DELREC~FILETYPE=xxxx,[KEYNO=n],KEY=(...)

This request deletes a record of the specified file type from the data base. The record is referenced by its key. The KEY field is mandatory, and you may use an optional KEYNO field if you are using a second key.

NOTE
One single **DELREC** transaction may be submitted at a time.

FILETYPE	A required keyword designating the file from which the record will be deleted.
KEYNO	An optional keyword for indexed files designating the key number to be used to identify the associated record to be deleted.
KEY	A required keyword for indexed files identifying the associated record to be deleted.

When the caller commits the transaction, the server performs syntax checking and data validation. The following validation tests are performed:

NOTE
The record to be deleted must already exist and not be locked by another user.

If no syntax or validation problems are encountered, the server will return a code of **0** and the record will be deleted. Otherwise, the transaction is rejected and an error code of **–5** is returned.

DISCON

Syntax:

DISCON

This command disconnects the client from the server. The client must reconnect to the TCP/IP socket connection.

EXECRW

Syntax:

EXECRW~RPTNAME=x,OUTPUTTYPE=FILE | PRINTER[,PROMPT=…],[PRINTER=x]

This request executes the specified Report Writer report and routes the output to the indicated destination.

The following keywords may be used:

RPTNAME	A required keyword representing the name of the report to be executed.
OUTPUTTYPE	A required keyword representing the destination output type, either ***FILE*** or ***PRINTER***.
PROMPT	An optional comma-delimited series of arguments *(enclosed in parentheses if there is more than one argument)* to provide responses to Report Writer prompts *(e.g. ask range, ask criteria)*. If an argument represents a key in a file*,* ***#F*** may be used to represent the first record and ***#L*** to represent the last record in the file respectively.
PRINTER	Keyword is required if ***OUTPUTTYPE*** is ***PRINTER***. The value supplied should be the name of the printer as known to the operating system.

If the report is successfully executed, and ***OUTPUTTYPE*** is ***FILE***, the server assigns a filename and places the output in that file. The name of the file is returned as the second field in the server response string. The ***GETDATAFILE*** command may then be used by the client to retrieve the report output.

Example:

```
EXECRW~RPTNAME=apvendor,OUTPUTTYPE=PRINTER,PRINTER=hplaser
```

Executes the Report Writer report named *apvendor* and directs output to the printer named *hplaser*.

EXECRW~RPTNAME= apvendor,PROMPT=(#F,#L),OUTPUTTYPE=FILE

Executes the Report Writer report named *apvendor* and directs output to a file. The report prompts for a beginning and ending range, and prompt information is supplied. If the report is successfully executed, the server response string might be something like:

0~rw38743

In this case, *rw38743* is the name of the server file containing the Report Writer output. The file name can be passed to *GETDATAFILE* to retrieve the report output.

GETCURRENT PERIOD

Syntax:

GETCURRENTPERIOD

This request returns the current accounting period based on the current system date.

Example:

GETCURRENTPERIOD

Returns the current accounting period. The server response string is always *0*.

See Also:

GETSYSDATE

SET

EXECSH

Syntax:

EXECSH~COMMAND={command} {parameters}

This command executes a shell script or program in the operating system.

FORMAT

Syntax:

FORMAT~DATAVALUE=X,FORMAT=X

This command can be used to format text, numbers or dates. There are four variations on this command. These are defined below:

DATE	You may format a date to include the slashes (01/01/24) as normally printed on reports. An example of a date format could be: FORMAT~DATAVALUE=010124,FORMAT=(%s^d^) The result would be 01/01/24.
MIDSTRING	You may use the midstring command to only print certain characters within a line of text. An example of a midstring format could be: FORMAT~DATAVALUE=This is the life, FORMAT=(%s^m13,4) The result would be life. The **“m13,4”** defines to start on the **13** character and include the next **4** characters.

MONETARIAL – STANDARD FORMAT	This command will allow you for format monetarial values to include dollar signs, commas and negative signs. FORMAT~DATAVALUE=1153.25,FORMAT=(%10.2f^$,-^) The result would be $ 1,153.25.
MONETARIAL – CURRENCY CODES	This command will allow you to print a dollar amount using a specific country's currency code. This country must be set up in **the Infinity POWER** system as a valid **"Country."** FORMAT~DATAVALUE=1153.25,FORMAT=(%10.2f~$,-[JDM]^) The result would be JDM J$ 1,153.25.

GETCUSTTAX

Syntax:

GETCUSTTAX~CUSTNO=...

For the specified customer, this request returns the tax rate total before the breakpoint, the breakpoint amount, the tax rate total after the breakpoint, and whether shipping is taxed. The following keyword is used:

CUSTNO	A required keyword designating the customer for which tax rate information is desired.

If the request succeeds, the second through fifth result fields contain the tax rate total before the breakpoint, the breakpoint amount, the tax rate total after the breakpoint, and whether shipping is taxed (Y or N) respectively.

Example:

GETCUSTTAX~CUSTNO=ALLE1

Get tax rate information for the customer "ALLE1." If the request succeeds, the result string is something like "0~6.0~5000.00~5.0~Y." The tax rate before the breakpoint is 6.0, the breakpoint is $5000.00, the tax rate after the breakpoint is 5.0, and shipping should be taxed.

GETDATAFILE

Syntax:

GETDATAFILE~FILENAME=x,OP=FIRST

GETDATAFILE~FILENAME=x,OP=NEXT

GETDATAFILE~FILENAME=x,OP=STOP

This request streams the contents of a server data file to the client. If data is successfully retrieved, the server result code is **"0;"** the streamed data is appended to the response string. For example, successful execution of the request:

GETDATAFILE~FILENAME=rw135,OP=FIRST

where the file *rw135* contains Report Writer output might produce an initial server response such as:

```
0~I N F I N I T Y  POWER  Sample  Company Inc    *  *  *  Sales    Order  *  *  *
```

Remaining data in the data file is retrieved with one or more invocations of the ***GETDATAFILE~FILENAME=x, OP=NEXT*** command, with a similar server response string. When no more data is available, an error code is returned. ***GETDATAFILE~FILENAME=x,OP=STOP*** closes the specified data file. If in the process of retrieving records any type of error occurs, the file is closed automatically.

The following keywords may be used:

FILENAME	A required keyword specifying the name of the file with contents to retrieve.
OP	***FIRST*** - Initiate retrieval at beginning of file. ***NEXT*** - Get next set of data ***STOP*** - Stop retrieval and close the file.

Examples:

GETDATAFILE~FILENAME=rw135, OP=FIRST

Get initial set of data from the file rw135.

GETDATAFILE ~FILENAME=rw135, OP=NEXT

Get next set of data from the file rw135.

GETDATAFILE ~FILENAME=rw135, OP=STOP

End data retrieval and close the file.

GETDB

Syntax:

GETDB

This request returns an integer code representing the installed database engine. The supported database engines and their corresponding codes are:

Engine	Code
CODEBASE	0
IBM DB2	1
MS SQL SERVER	2

Example:

GETDB

Returns an integer representing the installed database engine. The server response string is always **0**.

GETPRICE

Syntax:

GETPRICE~ITEMNO=...,CUSTNO=...[,QTY=...][,UM=...]

This request returns the sales price of an inventory item for a specified customer.

The following keywords may be used:

ITEMNO	A required keyword designating the inventory item of interest.
CUSTNO	A required keyword designating the customer whose price plan should be used to determine the price.
QTY	An optional keyword designating the quantity of items to use in determining the price. The quantity may contain a decimal fraction.
UM	An optional keyword designating the unit of measure to use in determining the price.

If the request succeeds, the second field of the result string contains the price.

Example:

GETPRICE~ITEMNO=00 100 10X12 BLT,CUSTNO=ALLE1

Get price for the inventory item **"00 100 10X12 BLT"** and customer **"ALLE1."** If the request succeeds, the result string is something like **"0~14.99."**

GETQTY

Syntax:

GETQTY~ITEMNO=...

This request returns the quantity available for a designated inventory item. The quantity available is calculated as the on-hand quantity minus the reserved quantity plus the quantity on order. The following keyword is used:

ITEMNO	A required keyword designating the inventory item number of the desired item.

If the request succeeds, the second field of the result string contains the quantity available.

Example:

GETQTY~ITEMNO=00 100 8X10 BLT

Get quantity available for the inventory item **"00 100 8X10 BLT."** If the request succeeds, the result string is something like **"0~13.5."**

GETREC

Syntax:

GETREC~FILETYPE=xxxx[,KEYNO=n][,KEY=...][,BEGKEY=...]

[,ENDKEY=...][,FIELDS=...][,MASK=...]

GETREC~NEXT

GETREC~STOP

This request accesses one or more records in the Data Pro data base and optionally returns the contents of those fields specified by the **FIELDS** ***keyword***.

If a specified record is successfully retrieved, the server result code is "**0**"; optional field values are appended to the response string *(tilde-delimited)* in the order specified by the **FIELDS** keyword. For example, successful execution of the request:

```
GETREC~FILETYPE=GL01,KEY=1112,FIELDS=(ACCOUNT,DESC)
```

produces a server response such as:

```
0~11120~Petty Cash
```

GETREC supports three types of file access: non-keyed, exact key, and range. Non-keyed access is intended for configuration files; the first record in the file is accessed. Exact key access retrieves a specific record in a keyed file based on designated key values. Range key access retrieves a range of records in a keyed file based on a key value range. Most keywords are appropriate to a particular type of file access.

The following keywords are applicable to all types of file access:

ITEMNO	A required keyword specifying the Data Pro file type to access. Example: **FILETYPE=GL01**
FIELDS	An optional keyword specifying one or more comma-delimited field names for which data should be returned to the client. Both standard Data Pro fields and user-defined fields may be specified. For fields consisting of array elements, use notation **"[n]"** after the field name to specify the desired array element. *(Array element origin is **1**.)* Examples: **FIELDS=DESC,FIELDS=(ACCOUNT,AMT[1])**
The following keyword is applicable only to keyed files:	
FIELDS	An optional keyword specifying one or more comma-delimited field names for which data should be returned to the client. Both standard Data Pro fields and user-defined fields may be specified. For fields consisting of array elements, use notation **"[n]"** after the field name to specify the desired array element. *(Array element origin is **1**.)*
KEYNO	An optional keyword that specifies which key to use for searching. If not specified, the default key number is **1**. Valid values are **1-10**. Example: **KEYNO=2**
The following keyword is applicable only to exact keyed file access:	
KEY	A required keyword that specifies the key values that uniquely identify a desired record. One value must be specified for each segment making up the key. Multiple values are comma-delimited and enclosed in parentheses. Examples: **KEY=11120**, **KEY=(091524,423).**
The following keywords are applicable only to range-keyed file access:	
BEGKEY	A required keyword specifying the key values of the first record in the range. Same syntax as the **KEY** keyword. In place of a key value, the notation **#F** may be used to represent the first key value in a key segment.
ENDKEY	A required keyword specifying the key values of the last record in the range. Same syntax as the **KEY** keyword. In place of a key value, the notation **#L** may be used to represent the last key value in a key segment.
The following keywords are applicable only to all file access:	

MASK	An optional keyword that may be used to screen records based on the value of the mask expression. The form of the mask expression is: **MASK=fldname1=expr1[^OP^fldname2=expr2]...** where **fldname1** is a database field name, **expr1** is a value to compare against the data value contained in the retrieved record, and **OP** is the logical operator **AND** or **OR**. **expr1** may contain one or more wildcards of the form ***** or **?,** where * represents a match of zero or more characters, and **?** matches a single character. **fldname1=expr1** is a relational expression; any number of relational expressions may be concatenated using the logical operators. **fldname1** must be a string or date type. Grouping of relational expressions is not supported. Examples: **MASK=ACCOUNT=11010,MASK=ACCOUNT=11???^AND^DESC=*Taxes***
NEXT	An optional keyword used on subsequent calls to **GETREC** to retrieve the next record in the range.
STOP	An optional keyword used on subsequent calls to **GETREC** to terminate record retrieval for the current range request.

Example:

GETREC~FILETYPE=GL01,KEY=11110,FIELDS=(ACCOUNT,DESC)

Get ACCOUNT and DESC fields from the GL01 file for account 11110. Key #1 is assumed.
GETREC~FILETYPE=AR02,BEGKEY=(HOME1,#F,501),ENDKEY=(HOME1,#L,#L),
FIELDS=(SLM,TAXNUM),MASK=DUEDATE=????24 GETREC~NEXT

The first **GETREC** request retrieves the first record in the specified range for which **DUEDATE** has the year **24**. Values for fields **SLM** and **TAXNUM** are returned to the client. The next **GETREC** request returns the next record in the range having the specified mask criteria. This example illustrates how a global variable representing the store configuration ID can be defined in a single document and referenced in other documents as needed.

GETRECSQL

Syntax:

GETRECSQL~FILETYPE=xxxx,

[KEYNO=n],

[BEGKEY=…],

[ENDKEY=…],

[FIELDS=(fldname1,fldname2,…)],

[ORDERBY=(KEYNO=n) | (FIELDS=fldname1,fldname2,…)],

[RELATION=(ID=x,FIELD=fldname,OP=EQ | NE | LT | LE | GT | GE,VALUE=x)]…,

[EXPR=x],

[MATCHFIELD=fldname],

[S_FILETYPE=xxxx],

[S_RELATION=(ID=x,FIELD= fldname,

OP=EQ | NE | LT | LE | GT | GE,VALUE=x)]…,

[S_EXPR=x],

[S_MATCHFIELD=fldname],

OP=COMMIT | ABORT

GETRECSQL~NEXT

GETRECSQL~STOP

This command is similar to ***GETREC***, but is applicable only to **SQL** databases. *(It is currently implemented only for MS SQL Server.)* ***GETRECSQL*** allows the construction of complex expressions for selecting data from a file, as well as the selection of records from a primary file based on the contents of a secondary file.

The ***GETDB*** command can be used to determine the currently active data base engine, and thus whether ***GETRECSQL*** is callable. If ***GETRECSQL*** is not callable, the server returns **-15** *(invalid keyword).*

GETRECSQL keywords may be entered as separate requests. The command is evaluated and executed upon receipt of the ***OP*** keyword.

The following keywords may be used:

FILEYPE	A required keyword specifying the Data Pro file type to access. This file is considered to be the primary file, that is, the file from which records are selected for output.
KEYNO	An optional keyword that specifies which key to use when specifying first and last record range constraints on the primary file Required if specifying the **BEGKEY** or **ENDKEY** keywords. This keyword is applicable only to indexed files.
BEGKEY	An optional keyword specifying the first record of a range of records to be considered for selection from the primary file. If not specified, the range begins with the first record of the file. For details on specifying a **BEGKEY** value, refer to the discussion on **BEGKEY** for the **GETREC** command. This keyword is applicable only to indexed files.
ENDKEY	An optional keyword specifying the last record of a range of records to be considered for selection from the primary file. If not specified, the range ends with the last record of the file. For details on specifying an **ENDKEY** value, refer to the discussion on **ENDKEY** for the **GETREC** command. This keyword is applicable only to indexed files.
ORDERBY	An optional keyword designating the database fields on which returned records should be sorted. Sort sequence is in ascending order. If not specified, records are returned in the order in which they are present in the primary file. There are two ways of specifying the sort order, represented by a choice of the following keywords:
KEYNO	Sort the output according to the database fields associated with the specified key number. This keyword is applicable only to indexed files.
FIELDS	A sequence of one or more database field names, indicating the fields on which to sort the returned results. The first field name is the most significant sort key. For details on specifying a **FIELDS** value, refer to the discussion on **FIELDS** for the **GETREC** command.

FIELDS	An optional sequence of one or more database field names to be output, specified in the order they should appear in the server result string. For details on specifying a ***FIELDS*** value, refer to the discussion on ***FIELDS*** for the ***GETREC*** command.
RELATION	An optional keyword specifying a criterion on which to select records from the primary file. A relation consists of a database field name, a comparison operator, and a value to be tested. Up to **50** relation keywords may be specified for the primary file. The relations are referenced in a corresponding ***EXPR*** keyword, which represents a logical expression. The logical expression designates which records should be selected from the primary data file. A relation consists of the following keywords, each of which is required:
ID	An up to **20**-character alphanumeric sequence uniquely identifying a specific relation. The **ID** is referenced in the ***EXPR*** keyword as part of a logical expression.
FIELD	The database field name to be tested. Specified as described for a field name in the ***FIELDS*** keyword of the ***GETREC*** command.
OP	One of the following relational operators to be used in comparing the database field name with a specified value: ***EQ*** **- equal to** ***NE*** **- not equal to** ***LT*** **- less than** ***LE*** **- less than or equal to** ***GT*** **- greater than** GE **- greater than or equal to**
VALUE	A data value to be compared with the specified database field name using the designated relational operator. The value must be compatible with the data dictionary type of the field name.
EXPR	An optional sequence of one or more relation **id's** separated by logical operators that together constitute a complete logical expression. Logical operators are **"AND"** and **"OR."** Partial logical expressions within the complete logical expression may be grouped using left and right brackets **("[" and "]", not parentheses!)**.
S_FILETYPE	An optional keyword designating the file type of a secondary file to be used as a basis for selecting records from the primary file. If ***S_FILETYPE*** is specified, the keywords ***MATCHFIELD*** and ***S_MATCHFIELD*** must also be specified.
MATCHFIELD	An optional database field name designating a field in the primary file that is to be used as a basis for including records. Only those field name values that match values in the database field of a secondary file as specified by the ***S_MATCHFIELD*** keyword are included. Format is as described for a field name in the ***FIELDS*** keyword of the ***GETREC*** command.

S_MATCHFIELD	An optional database field name designating a field in the secondary file that is to be used as a basis for selecting records from the primary file. Only those records in the primary file having field contents matching the contents of the designated field in the secondary file are selected. The ***MATCHFIELD*** keyword specifies the database field in the primary file and the ***S_MATCHFIELD*** keyword specifies the corresponding field in the secondary file that are to be used as the basis for selection. Format is as described for a field name in the ***FIELDS*** keyword of the ***GETREC*** command.
S_RELATION **S_EXPR**	Used to specify which records should be considered in the secondary file. Same functionality and format as described for the primary file keywords ***RELATION*** and ***EXPR***.
OP	A required keyword requesting the server to either process or terminate the ***GETRECSQL*** request. If ***COMMIT*** is specified and one or more records in the database meet the specified criteria, the server returns the first of the records. Subsequent records may be retrieved using the ***GETRECSQL NEXT*** keyword; the ***STOP*** keyword terminates an active sequence of returned records. If ***ABORT*** is specified, the ***GETRECSQL*** request is discarded.
NEXT	An optional keyword used on subsequent calls to ***GETRECSQL*** to retrieve the next record.
STOP	An optional keyword used on subsequent calls to ***GETRECSQL*** to terminate record retrieval.

Example:

```
GETRECSQL~FILETYPE=GL01

GETRECSQL~KEYNO=1

GETRECSQL~BEGKEY=30000

GETRECSQL~ENDKEY=#L

GETRECSQL~ORDERBY=(FIELDS=TYPE,ACCOUNT)

GETRECSQL~FIELDS=(ACCOUNT,DESC)

GETRECSQL~RELATION=(ID=TYPE_TEST,FIELD=TYPE,OP=EQ,VALUE=1)

GETRECSQL~RELATION=(ID=BUDGET_TEST,FIELD=BUDGET[1],OP=GT,VALUE=1000)

GETRECSQL~EXPR=TYPE_TEST AND BUDGET_TEST

GETRECSQL~S_FILETYPE=GL02

GETRECSQL~MATCHFIELD=ACCOUNT

GETRECSQL~S_MATCHFIELD=ACCOUNT

GETRECSQL~S_RELATION=(ID=AMT_TEST,FIELD=AMT,OP=GT,VALUE=500)

GETRECSQL~S_EXPR=AMT_TEST

GETRECSQL~OP=COMMIT
```

This request begins by considering only those records in secondary file **GL02** in which the ***AMT*** field is greater than 500. Only those records in primary file **GL01** having **1)** an ***ACCOUNT*** field that matches the ***ACCOUNT*** field in the selected secondary file records, **2)** having ***ACCOUNT*** values in the range 30000 and higher, and **3)** both an account type of asset and a first period budget amount greater than 1000 are retrieved.

The records are returned in order by account number within account type. Each server response string representing a returned record contains database fields in order by account number and account description.

See Also:

GETDB

GETREC

GETSOINFO

Syntax:

GETSOINFO~ORDERNO=x

This request provides status information for the requested sales order, including the order amount, weight, and shipping information.

The following keyword may be used:

ORDERNO	A required keyword designating the sales order number for which information is desired.

Field	Description
1	0 (result code)
2	Sales order amount
3	Weight total
4	Customer number
5	Customer name
6	Ship-To name
7	Ship-To address line 1
8	Ship-To address line 2
9	Ship-To city
10	Ship-To state
11	Ship-To zip code
12	Ship-To country
13	Terms
14	Shipping Instructions

Example:

GETSOINFO~ORDERNO=12345

Return status information for sales order 12345. If successful, the server response string is something like:

0~1345.35~123.35~9583~Alan Taylor~Rod Taylor~1355 Harbor Road~ Suite 150~Muttontown~New York~11238~US

GETSYSDATE

Syntax:

GETSYSDATE~TYPE=BEGMON | ENDMON | BEGYR | ENDYR | DATE

This request returns various types of system date information from the server.

The following keyword may be used:

TYPE	A required keyword designating the type of date information to return: **BEGMON** - Current month beginning date **ENDMON** - Current month ending date **BEGYR**- Current year beginning date **ENDYR**- Current year ending date **DATE** - Current system date

Example:

GETSYSDATE~TYPE=BEGMON

If the current system date is 043024, the server returns the response string 0~040124.

See Also:

GETCURRENTPERIOD

SET

LOGIN

Syntax:

LOGIN~USERNAME=x,PASSWORD=x

This request allows the client to log in as another user.

The following keywords may be used:

USERNAME	A required keyword designating the user name.
PASSWORD	A required keyword designating the user password.

Example:

LOGIN USERNAME=payroll,PASSWORD=djfi355

Logs in the client using the specified user name and password. If successful, the server response string is ***0***.

INVOICESO

Syntax:

INVOICESO ~ORDERNO=x

This request generates billing for the specified sales order.

The following keyword may be used:

ORDERNO	A required keyword designating the sales order number to be invoiced.

Example:

INVOICESO~ORDERNO=12345

Invoices sales order #12345.

PAYAR

Syntax:

PAYAR~OP=ADD,RECTYPE=HEADER,DATA=(item1,...)

PAYAR~OP=ADD,RECTYPE=DETAIL,DATA=(item1,...)

PAYAR~OP=COMMIT

PAYAR~OP=ABORT

This request posts an Accounts Receivable payment or deposit transaction to the database. The payment transactions may or may not be distributed to invoices. The deposit transaction may not be associated with any specific invoice. A deposit may not be taken for a **“Balance Forward”** customer.

To post an Accounts Receivable payment transaction, the client must assemble and transmit an Accounts Receivable payment header record and, optionally, from **0** to **4000** detail records. The header and detail records may be transmitted in any order. If a detail record has already been transmitted, the subsequent header record must be for a payment or an error will be returned. The server queues the submitted data until the client transmits a **“COMMIT”** operation; at that time, the server updates the accounting files.

Alternatively, the client may transmit an **“ABORT”** operation to release the queued data and thereby cancel posting of the transaction in progress. If the Accounts Receivable Transaction is successfully posted, the server returns the Accounts Receivable transaction number as the second field in the result string.

The following keywords may be used:

OP	A required keyword specifying the type of posting operation to perform. **ADD** specifies that a header or detail record is being transmitted as part of the current posting transaction. **COMMIT** specifies that the server should update its accounting files based on previously transmitted header and detail records. **ABORT** indicates that the server should cancel the transaction in progress.
RECTYPE	A keyword required when transmitting header or detail Accounts Receivable payment transaction data. Must be either ***HEADER*** or ***DETAIL***, representing the type of data being transmitted.
DATA	A keyword required when transmitting header or detail Accounts Receivable Transaction data. It contains the comma-delimited list of data items comprising the header or detail Accounts Receivable payment transaction record. If an item is not supplied, its position must still be represented by a comma.

The following data items comprise a **HEADER** record, and must be specified as key values in the order given below:

Key Value	Validation	Comments

A/R TRANSACTION TYPE INCLUSIVE	Integer from 7 – 8. Required. 7=Deposit, 8=Payment
CUSTOMER NUMBER	1 - 10 characters. Required. Must exist in the customer file. Balance Forward customer cannot have Deposit posted.
A/R PAYMENT DATE	Date in form *mmddyy*. Optional. If not specified, the system date is used.
A/R TRANSACTION NUMBER	1 - 10 characters. Required.
DESCRIPTION	1 - 20 characters. Optional. If not specified, the server assigns a description based on the order type.
TOTAL PAYMENT AMOUNT	Floating point number, Required. If payment, decimal fraction optional. Difference of applied and paid will be applied either to cash received or payment discounts.
A/R PAYMENT TYPE	Integer 1 – 10. Required. Must be valid payment type.
CREDIT CARD NUMBER	1 - 30 characters Required if Credit Card payment.
CREDIT CARD EXPIRATION	4 characters – numeric. Required if Credit Card payment.
CREDIT CARD HOLDER'S NAME	1 - 30 characters If Credit Card payment required. .
AUTHORIZATION NUMBER	1 - 6 characters. Required if payment type is greater than 2, non-EFT payment and not a dial-up payment code.
SALESMAN	1 - 5 characters. Optional.
PAYMENT SALES CODE	Alternate cash code. Optional. Determined by pay type.
A/R SALES CODE	Alternate A/R Account. Optional.
REGISTER NUMBER	1 - 3 characters. Optional. Point of Sale.

The following data items comprise a **DETAIL** record, and must be specified as key values in the order given below:

Key Value	Validation	Comments

CUSTOMER	1 - 10 characters. Required. Must exist in Customer File.
INVOICE NUMBER	1 - 10 characters. Required. Must exist in Open Item File.
INVOICE TYPE	Integer - range 1 to 7. Required. Needed to identify Open Item.
INVOICE DATE	6 characters. Required. Needed to identify Open Item.
PAYMENT AMOUNT	Floating point number. Required. May not be more than due on invoice.

Upon submitting a header or detail record, the server may detect a data validation error. The second field of the result string contains an error message in the form **"#:name:desc"** where # is the position of the data item in the ***DATA*** key value list, *name* is the name of the data item, and *desc* is a brief explanation of the problem. There may also be other error messages that apply to the header or detail record in general.

The client should check the server result string after submitting a ***COMMIT*** to verify that the Accounts Receivable Transaction was successfully posted. If successful, the second field of the result string contains the Accounts Receivable Transaction number.

Example:

```
PAYAR~OP=ADD,RECTYPE=HEADER,DATA=(,ALLE1,,1,,,,,,,,,,  5 ,,house,,net 30,1,10,,,,,)

PAYAR~OP=ADD,RECTYPE=DETAIL,DATA=(31100,`00  100 8x12 blt`,,,1,,,5.00,)

PAYAR~OP=COMMIT
```

Submits and posts an Accounts Receivable invoice with the following attributes -- Header record: the server assigns the Accounts Receivable Transaction number, customer *ALLE1*, normal sales order, tax number *5*, salesman *house*, terms *net 30*. Detail record: sales code *31100*, item number *100 8x12 blt*, 1 item ordered, unit price *5.00*. If the server successfully posts the transaction, the result string for the *COMMIT* is something like *0~10480*, where *10480* is the server-assigned Accounts Receivable Transaction number.

POSTAR

Syntax:

```
POSTAR~OP=ADD,RECTYPE=HEADER,DATA=(item1,...)
POSTAR~OP=ADD,RECTYPE=DETAIL,DATA=(item1,...)
POSTAR~OP=COMMIT
POSTAR~OP=ABORT
```

This request posts an Accounts Receivable transaction to the database. The transactions may be invoices or invoice related. Inventory information is automatically updated to reflect the transaction. Payments are posted through the **POSTDEP** command.

To post an Accounts Receivable transaction, the client must assemble and transmit an Accounts Receivable header record and, optionally, up to **450** detail records. The header and detail records may be transmitted in any order.

The server queues the submitted data until the client transmits a **"COMMIT"** operation. At that time, the server updates the accounting files. Alternatively, the client may transmit an **"ABORT"** operation to release the queued data and thereby cancel the posting of the transaction in progress. If the Accounts Receivable Transaction is successfully posted, the server returns the Accounts Receivable transaction number as the second field in the result string.

The following keywords may be used:

OP	A required keyword specifying the type of posting operation to perform. ***ADD*** specifies that a header or detail record is being transmitted as part of the current posting transaction. ***COMMIT*** specifies that the server should update its accounting files based on previously transmitted header and detail records. ***ABORT*** indicates that the server should cancel the transaction in progress.
RECTYPE	A keyword required when transmitting header or detail Accounts Receivable transaction data. Must be either ***HEADER*** or ***DETAIL***, representing the type of data being transmitted.
DATA	A keyword required when transmitting header or detail Accounts Receivable Transaction data. It contains the comma-delimited list of data items comprising the header or detail Accounts Receivable Transaction record. If an item is not supplied, its position must still be represented by a comma.

The following data items comprise a **HEADER** record, and must be specified as key values in the order given below:

Key Value	Validation	Comments

A/R TRANSACTION TYPE INCLUSIVE	Integer from 0 – 6. Required. 1=Invoice 2=Credit Memo 3=Debit Memo 4=Adjustment 5=Finance Charge 6=Retainage
CUSTOMER NUMBER	1 - 10 characters. Required. Must exist in the customer file.
A/R INVOICE DATE	Date in form *mmddyy*. Optional. If not specified, the system date is used.
A/R TRANSACTION #	1 - 10 characters. Optional. If not provided, the server uses Accounts Receivable. Transaction auto-numbering to assign an invoice number.
CUSTOMER PURCHASE	1 - 10 characters. Optional. Order number.
DESCRIPTION	1 - 20 characters. Optional. If not specified, the server assigns a description based on the Accounts Receivable transaction type.

SHIP-TO NAME	1 - 30 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO ADDRESS 1	1 - 30 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO ADDRESS 2	1 - 30 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO ADDRESS 3	1 - 30 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO ADDRESS 4	1 - 30 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO CITY	1 - 30 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO STATE	1 - 15 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO ZIP	1 - 10 characters. Optional. If not specified, the server assigns from the customer file.
SHIP-TO COUNTRY	1 - 10 characters. Optional. If not specified, the server assigns from the customer file.
TAX RATE NUMBER	Tax rate number 1 - 10 characters. Optional. If not specified, taken from the customer file. Must exist in the tax rate file.
DISCOUNT PER CENT	Point number, Optional. If not supplied, decimal fraction optional taken from the customer file.
SALESMAN	1 -5 characters. Optional. If not specified, taken from the customer file. Must exist in the salesman file.
TERMS	1 - 20 characters. Optional. If not supplied, taken from the customer file.
TERM TYPE INCLUSIVE	Integer from 0 - 2. Optional. Ignored if *Terms* is not specified. 0 = fixed due date, 1 = net days, 2 = net days end-of-month.
TERM DAYS	If *Term type* is 0, specify an actual due date, unless in the form of mmddyy. Otherwise, if not specified, an integer from 0 - 999. Optional.
FREIGHT AMOUNT	Floating point number. Optional. Decimal fraction optional.

A/R SALES CODE	1-10 characters. Optional. If not specified, taken from the customer file. If none used, the master A/R account will be posted. If present, must exist in the sales code file.
PROPOSAL FLAG	1 character. Optional. Defaults to no. Y = Yes for proposal.
SALES ORDER NUMBER	1 - 10 characters. Optional. May be blank.
ORDER DATE	Date in form *mmddyy*. Optional. If not specified, the system date is used.
REGISTER NUMBER	1 - 3 characters. Optional. Point-of-sale register ID. May be blank.

The following data items comprise a **DETAIL** record, and must be specified as key values in the order given below:

Key Value	Validation	Comments

SALES CODE	1 - 5 characters. Optional. If not specified, Sales Code is taken from the inventory item record. Must exist in the sales code file.
INVENTORY ITEM NUMBER	1 - 20 characters. Optional. If specified, must exist in the inventory item file.
DESCRIPTION	1 - 30 characters. Optional. If not specified, and Inventory item number is specified, taken from the first line of the inventory item description.
TAXABLE	*Y* or *N*. Optional. If not specified, taken from sales code file.
QUANTITY ORDERED	Floating point number, Required. Decimal fraction optional.
UNIT OF MEASURE	1 - 4 characters. Optional.
LINE ITEM DISCOUNT	Floating point number. Optional. Decimal fraction optional.
UNIT PRICE	Floating point number. Required. Decimal fraction optional.
SALES AMOUNT	Floating point number. Optional. If not specified, decimal fraction optional. The server computes the sales amount based on the unit price and quantity.
INVENTORY SERIAL #	1 - 20 characters. Optional. Invalid serial number will require adjustment later.

Upon submitting a header or detail record, the server may detect a data validation error. The second field of the result string contains an error message in the form **"#:name:desc"** where # is the position of the data item in the ***DATA*** key value list, ***name*** is the name of the data item, and ***desc*** is a brief explanation of the problem. There may also be other error messages that apply to the header or detail record in general.

The client should check the server result string after submitting a ***COMMIT*** to verify that the Accounts Receivable Transaction was successfully posted. If successful, the second field of the result string contains the Accounts Receivable Transaction number.

Example:

```
POSTAR~OP=ADD,RECTYPE=HEADER,DATA=(1,ALLE1,,,,,,,,,,,,,5,,HOUSE,net 30,,,,,,,,)
POSTAR~OP=ADD,RECTYPE=DETAIL,DATA=(,`00  100 8x12 blt`,,,1,,,5.00,,)
POSTAR~OP=COMMIT
```

Submits and posts an Accounts Receivable invoice with the following attributes --

HEADER RECORD	The server assigns the Invoice transaction type 1, Accounts Receivable Transaction number is automatic, customer *ALLE1*, tax number *5*, salesman *house*, terms *net 30*.
DETAIL RECORD	Item number *00 100 8x12 blt*, 1 item ordered, unit price *5.00*.

If the server successfully posts the transaction, the result string for the ***COMMIT*** is something like ***0~10480***, where ***10480*** is the assigned Accounts Receivable Transaction number.

POSTDEP

Syntax:

POSTDEP~ORDERNO=x,AMT=n.n,PAYTYPE=n[,NAME=x]

[,CARDNO=x][,EXPDATE=x]

This request posts a deposit transaction to the accounting data files. If the type of payment is a credit card, electronic draft capture processing is also performed.

The following keywords may be used:

ORDERNO	A required keyword designating the sales order number.
AMT	A required keyword designating the amount of the deposit.
PAYTYPE	A value 1-10 which defines type of payment. 1=cash, 2=check, 3-10=credit card
NAME	A keyword required for credit card payment types. Name of the card holder.
CARDNO	A keyword required for credit card payment types. Card credit number.
EXPDATE	A keyword required for credit card payment types. Credit card expiration date.

If the deposit is successfully posted, the server response is **"0."** In addition, for credit card transactions, the second field of the server response string contains authorization/reference information.

If the deposit is not successful, the server response will include in the second field, a decline response, and the third field will contain a more precise error from the Credit Card Processor.

Example:

```
POSTDEP~ORDERNO=1001,AMT=238.58,PAYTYPE=3,NAME=kenneth liss,
CARDNO=3984398403934,EXPDATE=0424
```

POSTIM

Syntax:

POSTIM~OP=POST,DATA=(item1,...)

This request posts an inventory transaction. If the inventory post is successful, the server returns **(0)** followed by the cost/quantity posted in the next field.

The following keywords must be used:

OP	A required keyword specifying that a posting operation is to be performed.
DATA	A keyword specifying the comma-delimited list of data items comprising the information required to post the transaction. If a data item is not supplied, its position must still be represented by a comma.

The following data items comprise an **POSTIM** record, and must be specified as key values in the order given below:

Item	Validation	Comments
ITEM NUMBER	1 - 20 characters	Item number in the database.
TYPE	Integer	Type of posting. See below for valid types.
QUANTITY	Floating point, decimal option	Quantity to be posted.
ORDER QUANTITY	Floating point, decimal option	Quantity ordered on PO transaction.
COST AMOUNT	Floating point, decimal option	Cost amount on a receipt.
SALE AMOUNT	Floating point, decimal option	For withdraw due to sale.
MODE	Integer	Used to designate special posting conditions. See below for valid modes.
	0-10 Characters	Vendor, Customer or Job Number for tracking purposes.
SERIAL NUMBER	0-20 Characters	For serialized items or items tracked in lots. Only checks for a passed value before attempting posting. Posting may fail due to non-existent serial number.
DESC	0-20 Characters	Misc. description contained in transaction record.
INVOICE NUMBER	0-10 Characters	Invoice reference number for Accounts Payable or Accounts Receivable.

Valid Types are:

PURCHASE ORDER	1
RECEIPT	2
SALES ORDER	3
WITHDRAWAL	4
RETURN	5

Valid Modes are:

NORMAL	1
NO HISTORY UPDATE	2
NO RESERVE QUANTITY UPDATE	3
NO ACTUAL UPDATE OF TRANSACTIONS	4
TRANSFER	5
JOB COST TRANSACTION	6
ADJUSTMENT	7
ASSEMBLY	8
DISASSEMBLY	9

Example:

POSTIM~OP=POST,DATA=(1,2,3,4,5,6,7,8,9,10,11)

Submits an inventory post transaction. The fields 1 through 11 represent the order of the fields previously described. If the server successfully posts the transaction, the result string for the **POSTIM** is **(0)** followed by the cost amount, if applicable.

POSTPO

Syntax:

POSTPO~OP=ADD,RECTYPE=HEADER,DATA=(item1,...)

POSTPO~OP=ADD,RECTYPE=DETAIL,DATA=(item1,...)

POSTPO~OP=COMMIT

POSTPO~OP=ABORT

This request posts a purchase order to the data base. Inventory information is automatically updated to reflect the transaction. To post a purchase order transaction, the client must assemble and transmit a purchase order header record and, optionally, up to 200 purchase order detail records. The header and detail records may be transmitted in any order. The server queues the submitted data until the client transmits a **"COMMIT"** operation; at that time, the server updates the accounting files.

Alternatively, the client may transmit an **"ABORT"** operation to release the queued data and thereby cancel posting of the transaction in progress. If the purchase order is successfully posted, the server returns the purchase order number as the second field in the result string.

The following keywords may be used:

PO	A required keyword specifying the type of posting operation to perform.
ADD	specifies that a header or detail record is being transmitted as part of the current posting transaction. **COMMIT** specifies that the server should update its accounting files based on previously transmitted header and detail records. **ABORT** indicates that the server should cancel the transaction in progress.
RECTYPE	A keyword required when transmitting header or detail purchase order data. Must be either **HEADER** or **DETAIL**, representing the type of data being transmitted.
DATA	A keyword required when transmitting header or detail purchase order data. It contains the comma-delimited list of data items comprising the header or detail purchase order record. If an item is not supplied, its position must still be represented by a comma.

The following data items comprise a **HEADER** record, and must be specified as key values in the order given below:

Item	Validation	Comments
VENDOR NUMBER	1-10 characters	Required. Must exist in the vendor file.
PURCHASE ORDER NUMBER	1-20 characters	Optional. If not provided, the server uses sales order auto-numbering to assign an order number
TYPE	1-3	Required. 1=normal 2=request for proposal 3=auto renewal.
DATE	Date in form mmddyy	Optional. If not specified, the system date is used.
REQUEST DATE	Date in form mmddyy	Optional.
LAST DATE	Date in form mmddyy	Optional.
TAX RATE NUMBER	1 - 5 characters	Optional. If not specified, taken from the **PO00** file.
DISCOUNT	Floating point number, decimal fraction optional	Optional.
SHIP-TO NAME	1-30 characters	Optional. If not specified, the server assigns **"Same."**
SHIP-TO ADDRESS 1	1-25 characters	Optional. If not specified, the server assigns **"Same."**
SHIP TO ADDRESS 2	1-25 characters	Optional. If Ship-to address1 is **"Same,"** the server clears this field
SHIP-TO ADDRESS 3	1-25 characters	Optional. If Ship-to address1 is **"Same,"** the server clears this field.

SHIP-TO ADDRESS 4	1-25 characters	Optional. If Ship-to address1 is “Same,” the server clears this field.
SHIP-TO CITY	1-15 characters	Optional. If Ship-to address1 is “Same,” the server clears this field.
SHIP-TO STATE	1-15 characters	Optional. If Ship-to address1 is “Same,” the server clears this field.
SHIP-TO ZIP	1-10 characters	Optional. If Ship-to address1 is “Same,” the server clears this field.
SHIP-TO-COUNTRY	1-15 characters	Optional. If Ship-to address1 is “Same,” the server clears this field.
SHIP VIA	1-20 characters	Optional.
DESC	1-20 characters	Optional.
REQUESTED BY	1-20 characters	Optional.
PO HOLD Y/N	Y or N	Optional. Default is N.
VENDOR SO NUMBER	1-10 characters	Optional.
MISC. STATUS	1-20 characters	Optional.
TERM TYPE	Integer from 0 - 2 inclusive	Optional. Ignored if Terms is not specified. 0=fixed due date, 1=net days, 2 = net days end-of-month
TERM DAYS	If Term type is 0,	Optional. Ignored if Terms is not specified.
TERMS	1-20 characters	Optional. If not supplied, taken from the vendor file.
DISCOUNT DAYS	0-31	Optional. Defaults to vendor file if not supplied.
PAYMENT DISC	Floating point number, decimal fraction optional.	Optional. Defaults to vendor file if not supplied.
CURRENCY COUNTRY	1-10 characters	Optional.
CURRENCY FACTOR	Floating point number, decimal fraction optional.	Optional.

The following data items comprise a **DETAIL** record, and must be specified as key values in the order given below:

Item	Validation	Comments
INVENTORY ITEM NUMBER	1-20 characters	Optional. If specified, must exist in the inventory item file.
SERIAL NUMBER	1-20 characters	Optional.
VENDOR PART NUMBER	1-20 characters	Optional
G/L ACCOUNT	1-20 characters	Optional. If specified, must exist in the Chart of Accounts file unless the account number begins with a **"."**. A leading **"."** in the account number indicates that this is a description only line.
DESCRIPTION	1-30 characters	Optional. If not specified, and Inventory item number is specified, taken from the first line of the inventory item description.
QUANTITY	Floating point number, decimal fraction optional	Required.
UNIT OF MEASURE	1-4 characters	Optional.
TAX Y/N	Y or N	Defaults to **N**.
UNIT COST	Floating point number, decimal fraction	Required.
LINE ITEM DISCOUNT	Floating point number, decimal fraction optional.	Optional.
LINE ITEM HOLD	Y or N	Defaults to **N**.
LINE ITEM SHIP DATE	Mmddyy	Optional.
VENDOR SO NUMBER	1-10 characters	Optional.

Upon submitting a header or detail record, the server may detect a data validation error. The second field of the result string contains an error message in the form **"#:name:desc"** where # is the position of the data item in the **DATA** key value list, ***name*** is the name of the data item, and ***desc*** is a brief explanation of the problem. There may also be other error messages that apply to the header or detail record in general.

The client should check the server result string after submitting a **COMMIT** to verify that the purchase order was successfully posted. If successful, the second field of the result string contains the purchase order number.

POSTSO

Syntax:

POSTSO~OP=ADD,RECTYPE=HEADER,DATA=(item1,...)

POSTSO~OP=ADD,RECTYPE=DETAIL,DATA=(item1,...)

POSTSO~OP=COMMIT

POSTSO~OP=ABORT

This request posts a sales order to the data base. Inventory information is automatically updated to reflect the transaction. To post a sales order transaction, the client must assemble and transmit a sales order header record and, optionally, up to 200 sales order detail records.

The header and detail records may be transmitted in any order. The server queues the submitted data until the client transmits a **"COMMIT"** operation; at that time, the server updates the accounting files. Alternatively, the client may transmit an **"ABORT"** operation to release the queued data and thereby cancel posting of the transaction in progress. If the sales order is successfully posted, the server returns the sales order number as the second field in the result string.

The following keywords may be used:

OP	A required keyword specifying the type of posting operation to perform.
ADD	Specifies that a header or detail record is being transmitted as part of the current posting transaction. **COMMIT** specifies that the server should update its accounting files based on previously transmitted header and detail records. **ABORT** indicates that the server should cancel the transaction in progress.
RECTYPE	A keyword required when transmitting header or detail sales order data. Must be either **HEADER** or **DETAIL**, representing the type of data being transmitted.
DATA	A keyword required when transmitting header or detail sales order data. It contains the comma-delimited list of data items comprising the header or detail sales order record. If an item is not supplied, its position must still be represented by a comma.

The following data items comprise a **HEADER** record, and must be specified as key values in the order given below:

Item	Validation	Comments
SALES ORDER NUMBER	1-10 characters	Optional. If not provided, the server uses sales order auto-numbering to assign an order number.
CUSTOMER NUMBER	1-10 characters	Required. Must exist in the customer file.
CUSTOMER PO NUMBER	1-10 characters	Optional.
SALES ORDER TYPE	Integer from 1-65	Required. 1=normal, 2=auto-cancel, inclusive. 3=special order, 4=drop shipment, 5=proposal, 6=return authorization.
DESCRIPTION	1 - 20 characters	Optional. If not specified, the server assigns a description based on the sales order type.
ORDER DATE	Date in form mmddyy	Optional. If not specified, the system date is used.
SHIP DATE	Date in form mmddyy	Optional.
CANCEL DATE	Date in form mmddyy	Optional.
SHIP-TO NAME	1 - 30 characters	Optional. If not specified, the server assigns **"Same."**
SHIP-TO ADDRESS 1	1-30 characters	Optional. If not specified, the server assigns **"Same."**

SHIP-TO ADDRESS 2	1-30 characters	Optional. If Ship-to address1 is **"Same,"** the server clears this field.
SHIP-TO CITY	1 - 30 characters	Optional. If Ship-to address1 is "**"Same,"** the server clears this field.
SHIP-TO STATE	1 - 15 characters	Optional. If Ship-to address1 is **"Same,"** the server clears this field.
SHIP-TO ZIP	1 - 10 characters	Optional. If Ship-to address1 is **"Same,"** the server clears this field.
SHIP-TO COUNTRY	1 – 15 characters	Optional. If Ship-to address1 is **"Same,"** the server clears this field.
MISCELLANEOUS STATUS	1 - 30 characters	Optional.
TAX RATE NUMBER	1 - 5 characters	Optional. If not specified, taken from the customer file. Must exist in the tax rate file.
PRICE LEVEL	Integer from 1 - 11	Optional. If not supplied, taken from the customer file.
DISCOUNT PERCENT	Floating point number, decimal fraction optional	Optional. If not supplied, taken from the customer file.
SALESMAN	1-5 characters	Optional. If not specified, taken from the customer file. Must exist in the salesman file.
TERMS	1 - 20 characters	Optional. If not supplied, taken from the customer file.
TERM TYPE	Integer from 0 - 2 inclusive	Optional. Ignored if Terms is not specified. 0=fixed due date, 1=net days, 2 = net days end-of-month
TERM DAYS	If Term type is 0, date in form mddyy. Otherwise, an integer from 0 - 999 inclusive	Optional. Ignored if Terms is not specified.
JOB NUMBER	1-10 characters	Optional.
SHIP VIA	1-20 characters	Optional.
FREIGHT AMOUNT	Floating point number, decimal fraction optional.	Optional.
DEPOSIT AMOUNT	Floating point number, decimal fraction optional.	Optional.
A/R SALES CODE	1-20 characters	Optional. If not specified, taken from the customer file. If present, must exist in the sales code file.
EMAIL ADDRESS	1-50 characters	Optional.
LOCATION NUMBER	1-20 characters	Optional.

The following data items comprise a **DETAIL** record, and must be specified as key values in the order given below:

Item	Validation	Comments
SALES CODE	1 - 10 characters	Optional. If not specified, sales code is taken from the inventory item record. Must exist in the sales code file.
INVENTORY ITEM NUMBER	1 - 20 characters	Optional. If specified, must exist in the inventory item file.
DESCRIPTION	1 - 30 characters	Optional. If not specified, and Inventory item number is specified, taken from the first line of the inventory item description.
TAXABLE	Y or N	Optional. If not specified, taken from sales code file.
QUANTITY ORDERED	Floating point number, decimal fraction optional	Required.
UNIT OF MEASURE	1 - 4 characters	Optional.
LINE ITEM DISCOUNT	Floating point number, decimal fraction optional.	Optional.
UNIT PRICE	Floating point number, decimal fraction optional.	Required.
SALES AMOUNT	Floating point number, decimal fraction optional.	Optional. If not specified, the server computes the sales amount based on the unit price and quantity.

Upon submitting a header or detail record, the server may detect a data validation error. The second field of the result string contains an error message in the form **"#:name:desc"** where # is the position of the data item in the **DATA** key value list, ***name*** is the name of the data item, and ***desc*** is a brief explanation of the problem. There may also be other error messages that apply to the header or detail record in general.

The client should check the server result string after submitting a **COMMIT** to verify that the sales order was successfully posted. If successful, the second field of the result string contains the sales order number.

Example:

```
POSTSO~OP=ADD,RECTYPE=HEADER,DATA=(,ALLE1,,1,,,,,,,,,,,  5 ,,house,,net 30,1,10,,,,,)

POSTSO~OP=ADD,RECTYPE=DETAIL,DATA=(31100,`00  100 8x12 blt`,,,1,,,5.00,)
```

POSTSO~OP=COMMIT

Submits and posts a sales order transaction with the following attributes – Header record: the server assigns the sales order number, customer **ALLE1**, normal sales order, tax number 5, salesman house, terms net 30. Detail record: sales code 31100, item number 00 100 8x12 blt, 1 item ordered, unit price 5.00. If the server successfully posts the transaction, the result string for the **COMMIT** is something like **0~10480**, where **10480** is the server-assigned sales order number.

POSTVCH

Syntax:

POSTVCH~OP=ADD,RECTYPE=HEADER,DATA=(item1,...)

POSTVCH~OP=ADD,RECTYPE=DETAIL,DATA=(item1,...)

POSTVCH~OP=COMMIT

POSTVCH~OP=ABORT

This request posts an Accounts Payable voucher to the database. When a zero total header is entered and the detail represents a balanced General Ledger journal entry, no voucher is generated, but the posting to the General Ledger is made. The Vendor invoice count is updated.

To post an Accounts Payable voucher transaction, the client must assemble and transmit an Accounts Payable voucher header record and, optionally, up to **50** Accounts Payable voucher detail records. The header and detail records may be transmitted in any order. The server queues the submitted data until the client transmits a **"COMMIT"** operation.; at that time, the server updates the accounting files. Alternatively, the client may transmit an **"ABORT"** operation to release the queued data and thereby cancel posting of the transaction in progress. If the Accounts Payable voucher is successfully posted, the server returns the Accounts Payable voucher number as the second field in the result string.

The following keywords may be used:

OP	A required keyword specifying the type of posting operation to perform.
ADD	Specifies that a header or detail record is being transmitted as part of the current posting transaction. ***COMMIT*** specifies that the server should update its accounting files based on previously transmitted header and detail records. ***ABORT*** indicates that the server should cancel the transaction in progress.
RECTYPE	A keyword required when transmitting header or detail Accounts Payable voucher data. Must be either ***HEADER*** or ***DETAIL***, representing the type of data being transmitted.
DATA	A keyword required when transmitting header or detail Accounts Payable voucher data. It contains the comma-delimited list of data items comprising the header or detail Accounts Payable voucher record. If an item is not supplied, its position must still be represented by a comma.

The following data items comprise a **HEADER** record, and must be specified as key values in the order given below:

Item	Validation	Comments
VENDOR NUMBER	1 - 10 characters	Required. Must exist in vendor file.
A/P VOUCHER TYPE	Integer from 1-8 inclusive	Required. 1=Invoice, 2=Credit Memo, 3=Debit Memo, 4=Adjustment, 5=Statement, 6=Finance Charge, 7=Retainage, 8=Deposit/Advance
VOUCHER DATE	Date in form mmddyy	Required.
A/P VOUCHER NUMBER	1-10 characters	Required.
DESCRIPTION	1 - 20 characters	Optional. If not specified, the server assigns a description based on the A/P voucher type.
SALES AMOUNT	Floating point number, decimal fraction optional	Required.
TOTAL AMOUNT	Floating point number, decimal fraction optional	Required.
DISCOUNT PERCENT	Floating point number, decimal fraction optional	Optional. If not supplied, taken from the vendor file.
TERM TYPE	Integer from 1-5 inclusive	Optional. If not supplied, taken from the vendor file. 1=Immediate, 2=Net Days, 3=Net EOM, 4=Batch, 5=Suspense
TERM DAYS	Integer from 0-999	Optional. If not supplied, taken from the vendor file.
BANK ACCOUNT	1-5 characters	Required. Must exist in database.

The following data items comprise a **DETAIL** record, and must be specified as key values in the order given below:

Item	Validation	Comments
GL ACCOUNT	1 - 20 characters	Required. Must exist in database.
AMOUNT	Floating point number, decimal fraction optional	Required.

NOTE

When entering a credit memo, the server will automatically reverse the amounts entered.

Upon submitting a header or detail record, the server may detect a data validation error. The second field of the result string contains an error message in the form **"#:name:desc"** where # is the position of the data item in the ***DATA*** key value list, ***name*** is the name of the data item, and ***desc*** is a brief explanation of the problem. There may also be other error messages that apply to the header or detail record in general.

Possible errors:

Header:

Vendor does not exist.

Voucher already exists.

Detail:

GL Account does not exist

Commit:

-10 –1 Amount of distributions does not match header total. Difference returned in third position and Voucher Number is returned in fourth position.

-10 –2 For a zero dollar voucher, amount of distributions does equal zero. Difference returned in third position and Voucher Number is returned in fourth position.

Voucher already exists. – It may have been entered in the interim.

The client should check the server result string after submitting a ***COMMIT*** to verify that the Accounts Payable voucher was successfully posted. If successful, the second field of the result string contains the Accounts Payable voucher number.

Example:

POSTVCH~OP=ADD,RECTYPE=HEADER,DATA=(BOW1,1,120199,INV004,,1225.1,1225.1,0,0,0,)

POSTVCH~OP=ADD,RECTYPE=DETAIL,DATA=(65000,1225.1)

POSTVCH~OP=COMMIT

Submits and posts a Accounts Payable voucher transaction with the following attributes:

The Invoice Voucher # INV004 from vendor BOW1 for $1225.10 is posted to the open item file. Terms and bank account will be determined from vendor BOW1's record in the database. An entry to the General Ledger is posted to the integration file. It consists of a debit to account 65000 for 1225.10 and a credit to the A/P account specified in the bank account record specified in vendor BOW1's master record.

PRINTPO

Syntax:

PRINTPO~ORDERNO=x,FORM=x,PRINTER=x,TYPE=x

This request prints a purchase order using the designated form and printer.

The following keywords may be used:

ORDERNO	A required keyword designating the purchase order number.
FORM	A required keyword designating the name of the form file *(without the .frm extension).*
PRINTER	A required keyword designating the name of the printer to use. This is the name of the printer specified in the Printers option of the Data Pro System Administrator. The selected printer must have an output type of **1** *(printer device).*
TYPE	Type is an optional keyword designating the type of form that is being printed. Valid types are Order, RFP. The default type is Order.

Example:

PRINTPO~ORDERNO=10480,FORM=POORD1,PRINTER=HPLASER

This request prints the sales order 10480 using the form POORD1 and printer HPLASER.

PRINTSO

Syntax:

PRINTSO~ORDERNO=x,FORM=x,PRINTER=x

This request prints a sales order using the designated form and printer.

The following keywords may be used:

ORDERNO	A required keyword designating the sales order number.
FORM	A required keyword designating the name of the form file *(without the .frm extension).*
PRINTER	A required keyword designating the name of the printer to use. This is the name of the printer specified in the Printers option of the Data Pro System Administrator. The selected printer must have an output type of **1** *(printer device).*
TYPE	Type is an optional keyword designating the type of form that is being printed. Valid types are Order, Packing, Lading or Invoice. The default type is Order.

Example:

PRINTSO~ORDERNO=10480,FORM=SOORD1,PRINTER=HPLASER

This request prints the sales order 10480 using the form SOORD1 and printer HPLASER.

RECEIVEPO

Syntax:

RECEIVEPO~OP=RECEIVE,DATA=(PO, VSO, QT4, ITEM, LINE, VITEM, UM, SERIALNUM)) (To receive Inventory Items)

OR

RECEIVEPO~OP=VERIFY,DATA=(PO, VSO, QT4, ITEM, LINE, VITEM, UM, SERIALNUM) (To verify PO Line item)

OP=VERIFY - Returns **0** or error. If the **VERIFY** is successful, these values are also returned:

PO Number:

- **Vendor SO**
- **Vendor number**
- **Quantity to receive**
- **Inventory number**

NOTE

Quantity verification takes in to account the flag in **PO00** to not allow over receipt and the Unit of measure being received. If no **UM** is provided, the **UM** of the PO line item is used.

OP=RECEIVE - Returns 0 or error

Returns same info as verify, but also does necessary posting to Purchase Order inventory, etc.

The user must provide either a Purchase Order number or a vendor's sales order number. If a vendor's sales order is provided, the first matching vendor's sales number is used. The verify function will return the Vendor, and Purchase Order number. If it is not the correct Purchase Order, the Purchase Order number will have to be provided. If there is any chance of a duplicate in this case, the Purchase Order number should be used.

The user may provide either an inventory item number, a vendor's item number or a line number to identify the receipt. If more than one item is provided the following hierarchy is used:

- **Line number**
- **Inventory item number**
- **Vendor's item number**

The following field descriptions may be used:

PO	Purchase Order Number
VSO	Vendor Sales Order Number
QTY	Quantity to Receive
ITEM	Inventory Item Number
LINE	Purchase Order Line Number
VITEM	Vendor’s Item Number
UM	Unit of Measure
SERIALNUM	Serial Number as Lot Number

Example:

RECEIVEPO~OP=RECEIVE,DATA=(1001,,3.0,99-1001,1,,EA,,)

This request receives a quantity of 3 for the inventory item 99-1001 on line 1 of purchase order number 1001 and unit of measure each.

RECSOSHIP

Syntax:

RECSOSHIP~ORDERNO=x[,FRAMT=n.n][,SHIPINST=x]

[,TRACKNO=x]

This request records the shipment of the items in a sales order.

The following keywords may be used:

ORDERNO	A required keyword designating the sales order number.
FRAMT	An optional keyword designating the freight amount.
SHIPINST	An optional keyword describing the freight carrier and handling. *(e.g. FedEx Priority Overnite.)* May be a maximum of **20** characters.
TRACKNO	An optional keyword containing tracking information. May be a maximum of **30** characters.

Example:

RECSOSHIP~ORDERNO=12345,FRAMT=35.50,SHIPINST=Fed Exp,TRACKNO=7593

This request records shipment of items for sales order 12345. The freight amount is 35.50, the shipping instruction is Fed Exp, and the tracking number is 7593.

RPTBALSH

Syntax:

RPTBALSH ~FORMAT=x,

TYPE=NORMAL | BUDCOMP | COMPYR | COMPMOYR | 6PERTR,

OUTPUTTYPE=FILE | PRINTER

[,ACCTMASK=x]

[,PRINTER=x]

This request generates a balance sheet report and routes the output to the indicated destination.

The following keywords may be used:

FORMAT	A required keyword representing the specific balance sheet format to be used.
TYPE	A required keyword designating the type of balance sheet to be generated: **NORMAL** - Normal **BUDCOMP**- Budget comparison **COMPYR** - Comparative year **COMPMOYR** - Comparative month/year **6PERTR** - 6 period trend
OUTPUTTYPE	A required keyword representing the destination output type, either *FILE* or *PRINTER*.
ACCTMASK	An optional keyword representing an account mask, where the character **?** represents any character.
PRINTER	Keyword is required if ***OUTPUTTYPE*** is ***PRINTER***. The value supplied should be the name of the printer as known to the operating system.

If the report is successfully executed, and ***OUTPUTTYPE*** is ***FILE***, the server assigns a filename and places the output in that file. The name of the file is returned as the second field in the server response string. The ***GETDATAFILE*** command may then be used by the client to retrieve the report output.

Example:

RPTBALSH~FORMAT=balance,TYPE=normal,OUTPUTTYPE=PRINTER,PRINTER=hplaser

Generates a normal balance sheet using the format *balance* directing output to the printer named *hplaser*.

See Also:

GETDATAFILE

RPTINCST

Syntax:

RPTINCST ~FORMAT=x,

TYPE=NORMAL | BUDCOMP | COMPYR | BUDYRYR | 6PERTR | QTR | YTD |

YRBUD,

OUTPUTTYPE=FILE | PRINTER

[,ACCTMASK=x]

[,PRINTER=x]

This request generates an income statement and routes the output to the indicated destination.

The following keywords may be used:

FORMAT	A required keyword representing the specific income statement format to be used.
TYPE	A required keyword designating the type of income statement to be generated: NORMAL - Normal BUDCOMP - Budget comparison COMPYR - Comparative year BUDYRYR- Budget/year-to-year 6PERTR- 6 period trend QTR - Quarterly statement YTD-Year-to-date only YRBUD - Annual budget BUDCURYTD- Budget/Current & YTD
OUTPUTTYPE	A required keyword representing the destination output type, either ***FILE*** or ***PRINTER***.
ACCTMASK	An optional keyword representing an account mask, where the character ***?*** represents any character.
PRINTER	Keyword is required if ***OUTPUTTYPE*** is ***PRINTER***. The value supplied should be the name of the printer as known to the operating system.

If the report is successfully executed, and ***OUTPUTTYPE*** is ***FILE***, the server assigns a filename and places the output in that file. The name of the file is returned as the second field in the server response string. The ***GETDATAFILE*** command may then be used by the client to retrieve the report output.

Example:

RPTINCST~FORMAT=income,TYPE=compyr,ACCTMASK=???????999,
OUTPUTTYPE=PRINTER,PRINTER=hplaser

Generates a comparative year income statement using the format ***balance*** directing output to the printer named ***hplaser***.

See Also:

GETDATAFILE

SET

Syntax:

SET[~COMPANY=xxx[,DATE=mmddyy]

This request defines the database state to be used by subsequent requests. This must be the first command for each client/server connection. Each connection will initiate a new **POWERServer** environment based on the default set up for the **User ID** that **POWERServer** logs in as.

The following keywords may be used:

COMPANY	An optional keyword that specifies the company to use for data base access. Example: **COMPANY=ins.**
DATE	An optional keyword that specifies the accounting date to use for data base access. The date may be specified in the form **MMDDYY** or else as the literal **CLOCK**, in which case the server's current date is used. Examples: **DATE=040524, DATE=CLOCK.**

NOTE

If more than one keyword in a single **SET** request contains errors, the result code reflects the first keyword for which errors were encountered.

Example:

SET~COMPANY=ins,DATE=081824

Set company to "ins" and accounting date to "081824".

SET~COMPANY=dpa

Set company to **"dpa".** (Accounting date not affected)

SHIPSO

Syntax:

SHIPSO~OP=SHIP,RECTYPE=HEADER,DATA=(item1,...)

SHIPSO~OP=SHIP,RECTYPE=DETAIL,DATA=(item1,...)

SHIPSO~OP=COMMIT

SHIPSO~OP=ABORT

This request posts a sales order shipment to the database. Inventory information is automatically updated to reflect the transaction. To post a sales order shipment transaction, the client must assemble and transmit a sales order header record and, optionally, up to 450 sales order detail records. The header must be transmitted and validated before any detail records may be transmitted. The detail records are validated as they are submitted.

The server queues the submitted data until the client transmits a **"COMMIT"** operation; at that time, the server updates the accounting files. Alternatively, the client may transmit an **"ABORT"** operation to release the queued data and thereby cancel posting of the transaction in progress. If the sales order is successfully posted, the server returns **0**.

The following keywords may be used:

OP	A required keyword specifying the type of posting operation to perform. ***SHIP*** specifies that a header or detail record is being transmitted as part of the current posting transaction. ***COMMIT*** specifies that the server should update its accounting files based on previously transmitted header and detail records. ***ABORT*** indicates that the server should cancel the transaction in progress.
RECTYPE	A keyword required when transmitting header or detail sales order data. Must be either ***HEADER*** or ***DETAIL***, representing the type of data being transmitted.
DATA	A keyword required when transmitting header or detail sales order data. It contains the comma-delimited list of data items comprising the header or detail sales order record. If an item is not supplied, its position must still be represented by a comma.

The following data items comprise a **HEADER** record, and must be specified as key values in the order given below:

Item	**Validation**	**Comments**
SALES ORDER NUMBER	1 – 10 characters	Mandatory. Must exist in the database, must have items left to ship and must not be on hold.
FREIGHT AMOUNT	Floating point number, decimal fraction optional.	Optional. Replaces current value if entered.
SHIP VIA	1 - 20 characters.	Optional.
MISCELLANEOUS STATUS	1 - 30 characters.	Optional. May be used for a tracking number or other information.

The following data items comprise a **DETAIL** record, and must be specified as key values in the order given below:

Item	**Validation**	**Comments**
LINE NUMBER	Integer	Required. Must be a valid line number between 1 and 450 which is not on hold. May or may not be over-shipped based on the company configuration.
QUANTITY TO SHIP	Floating point number, decimal fraction optional	Required.
SERIAL/LOT NUMBER	Twenty **(20)** characters.	Optional. This field may be used to specify the serial number or lot number of the item being shipped.

Upon submitting a header or detail record, the server may detect a data validation error. The second field of the result string contains an error message in the form **"#:name:desc"** where # is the position of the data item in the ***DATA*** key value list, ***name*** is the name of the data item, and ***desc*** is a brief explanation of the problem. There may also be other error messages that apply to the header or detail record in general.

The client should check the server result string after submitting a ***COMMIT*** to verify that the sales order shipment was successfully posted. If successful, the return code will be **0**.

Example:

```
SHIPSO~OP=SHIP,RECTYPE=HEADER,DATA=(15840,123.45,`Ship via UPS`,B071498A)

SHIPSO~OP=SHIP,RECTYPE=DETAIL,DATA=(3,15)
```

SHIPSO~OP=COMMIT

Submits and posts a sales order shipment transaction with the following attributes –

Header record: Sales order 15840 was shipped with a total freight charge of $123.45. It was shipped via UPS Ground and the tracking number is B071498A.

Detail record: Fifteen Units of Line item three were shipped. If the server successfully posts the transaction, the result string for the ***COMMIT*** is **0**.

UPDREC

Syntax:

UPDREC~FILETYPE=xxxx,FIELDS=(...),DATA=(...)

[,LOCKREC=(FILETYPE=xxxx[,KEYNO=n][,KEY=(...)])]

[,KEYNO=n][,KEY=(...)]

[,CHECKMOD=(mmddyy,hh:mm:ss),OP=x

This request updates a record of the specified file type with one or more data values corresponding to the specified data base field names.

A single **UPDREC** transaction may be submitted as a single request or as a series of requests to the server. The server recognizes completion of the transaction when it receives the **OP** *(operation)* keyword with the key value **COMMIT**. At that time, the server processes the entire **UPDREC** transaction and performs the requested operation.

If a single **UPDREC** transaction is submitted as a series of requests, the calling program must observe a number of rules. Each keyword must be contained entirely within the request *(i.e., the keyword cannot be broken across requests)*. With the exception of the **OP** keyword *(which is present on the final submission)*, keywords may be specified in any order. The **DATA** and **FIELD** keywords may be specified multiple times within the transaction. If this is the case, the server treats the multiple occurrences as a single occurrence by concatenating them in the order in which they were submitted. The caller should check the server response after each submission for a successful result code. Because the server does not actually process the transaction until receiving a **COMMIT** operation, the final response string contains explanatory information if there is an error.

The following keywords may be used:

FILETYPE	A required keyword designating the file containing the record to be updated.
FIELDS	A required keyword consisting of one or more comma-delimited data base field names for the data to be updated in the record. The **DATA** keyword must contain a value for each field name specified.
DATA	A required keyword consisting of one or more comma-delimited data values to be included in the record. The corresponding **FIELDS** keyword must contain a field name for each data value specified.
KEY	A keyword required for indexed files identifying the record to be updated.
KEYNO	An optional keyword applicable to indexed files identifying the key to be used when identifying the record.
OP	A keyword that is specified at the time the server should actually process the transaction. May have the value **COMMIT** *(process the transaction)* or **ABORT** *(cancel the transaction).*
LOCKREC	An optional keyword that specifies an associated record to be locked while the **UPDREC** takes place. **LOCKREC** may contain the following keywords:
FILETYPE	A required keyword designating the file type of the associated record to be locked.
KEYNO	An optional keyword for indexed files designating the key number to be used to identify the associated record to be locked.
KEY	A required keyword for indexed files identifying the associated record to be locked.
CHECKMOD	An optional keyword consisting of two key values: a date in the form mmddyy and a time in the form **hh:mm:ss**. If the date and time in the record to be updated do not match the specified date and time, the server rejects the transaction with the assumption that the submitted data may be “stale.”

Because the **GETREC** request cannot lock a record, it is possible that data retrieved by **GETREC** may have been modified by another program between the time of the **GETREC** and the **UPDREC**. By retrieving the last modification date and time during the **GETREC** and passing them to the **CHECKMOD** keyword during an **UPDREC**, the caller can prevent updating a record with non-current data.

When the caller commits the transaction, the server performs syntax checking and data validation. The following validation tests are performed:

- The record to be updated must exist and cannot be in use by another program.
- Each data base field name must exist in the data base.
- The data types of provided data values must match the type of their associated data dictionary definition *(i.e., an alphanumeric string cannot be provided for a numeric value).*
- Additional file and field-specific **“reasonableness”** tests.

NOTE

If no syntax or validation problems are encountered, the record is updated. Otherwise, the transaction is rejected.

Example:

```
UPDREC~FILETYPE=GL00,FIELDS=(retain,intercopro),DATA=(15000,y,),OP=COMMIT
```

Update the GL00 record.

```
GETREC~FILETYPE=SO02,KEY=(10001,6),FIELDS=(modifydate,modifytime)
          ....
          ....
```

```
UPDREC~FILETYPE=SO02,LOCKREC=(FILETYPE=SO01,KEY=10001)
UPDREC~CHECKMOD=(date,time)
UPDREC~FIELDS=(holdyn,sc)
UPDREC~DATA=(n,10000)
UPDREC~FIELDS=(desc)
UPDREC~DATA=(`new and improved widget`)
UPDREC~OP=COMMIT
```

Update an **SO02** detail record. Lock the **SO01** record during the update. During the update, check to ensure that the record has not been modified by another program since the **GETREC**. *(NOTE: The calling program must set date and time to the values contained in the data base fields **modifydate** and **modifytime**.)*

ERROR CODES

This section lists error codes that can be returned by the server. Error codes are numbered and grouped into the following ranges by category:

Category	Range	Description
APPLICATION	-0001 through -5999	Application error.
SYSTEM	-6001 through -9999	Internal server error.

The error codes are as follows:

Error #	Applicable Requests	Description	Information Fields
-1	All	Keyword/key value invalid format	
-2	DELDATAFILE GETCUSTTAX GETDATAFILE GETQTY GETPRICE GETSOINFO INVOICESO POSTDEP POSTSO GETREC ADDREC UPDREC RECSOSHIP CRFILEDEF DELFILEDEF CRFILE DELFILE EXECRW RFTBALSH RFTINCST LOGIN CHECKRIGHTS GETSYSDATE	Missing keyword	First information field contains the name of the missing keyword.

-3		Unknown request type.	
-4	GETCUSTTAX GETDATAFILE GETQTY GETPRICE GETSOINFO INVOICESO POSTDEP POSTSO GETREC ADDREC UPDREC RECSOSHIP CRFILEDEF DELFILEDEF DELDATAFILE EXECRW RPTBALSH RPTINCST	Unable to open file.	If not **GETDATAFILE** or **DELDATAFILE**, the first information field contains file type. Otherwise, information field contains file name.
-5	GETCUSTTAX GETQTY GETPRICE POSTDEP POSTIM POSTSO GETREC ADDREC INVOICESO UPDREC GETSOINFO RECSOSHIP SOSHIP DELDATAFILE GETRECSQL	Unable to get record.	If **GETDATAFILE** information field contains file name, first information field contains file type. If keyed file, second information field contains key value if available.

-6	SET GETQTY GETPRICE POSTDEP POSTSO GETREC ADDREC UPDREC GETSOINFO RECSOSHIP GETSYSDATE CHECKRIGHTS LOGIN INVOICESO CRFILEDEF CRFILE DELFILE DEF DELFILE GETDATAFILE DELDATAFILE EXECRW RPTBLASH RPTINCST	Missing data	First information field contains the keyword for which data is missing. For **POSTSO**, **ADDREC**, and **UPDREC**, the second information field contains the name of the data base field if keyword is **DATA**. For **ADDREC** and **UPDREC**, contains name of a required data base field if the keyword is **FIELDS.**

-7	SET GETPRICE POSTDEP POSTSO GETREC ADDREC GETSOINFO RECSOSHIP UPDREC CRFILEDEF DELFILEDEF CRFILE DELFILE GETDATAFILE EXECRW RPTBALSH RPTINCST GETSYSDATE GETRECSQL	Invalid data.	First information field contains the keyword. Second information field contains the invalid data item. Third information field contains a description of the problem. For **POSTSO**, **ADDREC**, and **UPDREC, UPDREC**, the fourth information field contains the name of the data base field that is invalid.
-8	POSTDEP	Draft capture failed.	
-9	POSTDEP ADDREC UPDREC RECSOSHIP	Lock error.	

-10	All	Request-specific error.	First information field is the error subcode. Refer to Error Subcodes By Request Type Section for details.
-11	POSTSO ADDREC	Record already exists.	
-12	UPDREC	Required field not in data dictionary.	First information field is the data base field name.
-13	DELDATAFILE EXECRW RPTBALSH RPTINCST LOGIN CHECKRIGHTS	Operation failed.	Requested operation failed. First information field is a description of the operation that failed.
-14	GETDATAFILE	Fatal I/O Error	File name encountering error.
-15	GETRECSQL	Invalid key word.	Invalid keyword.

ERROR SUB-CODES BY REQUEST TYPE

Certain request types may return error conditions that are too specific for classification under one of the error codes listed in the **ERROR CODE** section on the previous pages. Such error conditions are classified as request-specific errors. The first information field of a request-specific error contains a numeric sub-code identifying the request-specific error condition.

The error sub-codes are as follows:

Request Type	Sub Code #	Description	Information Fields
PAYAR	-1	Maximum detail Lines exceeded	
	-2	Transaction not in progress	
	-3	Header record missing.	
	-4	Sales Code not set up for Cash Receipts	
	-5	Sales Code not set up for Payment Type	
	-6	Customer on Header or Detail is not same Customer entered on earlier record**.**	
	-7	Attempt to apply more than the amount of invoice.	
	-8	Open Item does not exist	
	-9	Amount to be distributed greater than available.	
	-10	CC not approved or EFT not set up.	
POSTAR	-1	Maximum detail lines exceeded	
	-2	Transaction not in progress	
	-3	Header record missing.	
	-4	Inventory item and inventory not integrated	-1
POSTIM	-1	No serial number submitted on serialized or lot item.	

POSTSO	-2	Transaction not in progress.	
	-3	Header record missing.	
GETREC	-1	No file is currently active.	
	-2	Range request already in progress.	
ADDREC	-1	No more memory to add new data value.	The first information field contains the keyword for which the data value cannot be added
	-2	Key field missing.	The first information field contains the primary key number. The second information field contains the data base name of the missing key field.
	-3	Unable to assign customer number	
SOSHIP	-1	More than the maximum number of items allowed.	
	-2	Attempt to commit when no transaction in progress.	
	-3	Attempt to add detail item when no valid header record has been received.	
	-4	Sales Order is fully shipped.	
	-5	Sales Order is on Hold.	
	-6	Line Item is on Hold.	
	-7	Attempt to send another header when a valid transaction is in process	
	-8	Description line only	
	-9	Line Item is over-shipped.	
UPDREC	-1	No more memory to add new data value.	The first information field contains the keyword for which the data value cannot be added.
	-2	Request modification time different from record modification time.	

GETRECSQL	-1	No file is currently active.	
	-2	Range request already in progress.	
	-3	Duplicate keyword.	First information field contains the duplicate keyword.
	-4	Maximum relations exceeded.	
CRFILEDEF	-1	Existing filetype	
	-2	Existing pattern	
	-3	Maximum fields exceeded	
	-4	Index, field name not found	Missing field name
	-5	Field array error	Invalid field name
DELFILEDEF	-1	Filetype not found	File type
INVOICESO	-1	Invalid sales order type	
	-2	Invoicing not performed	