Medicare Direct Entry (MDE) & Eligibility Edits

Overview guide

Last update: January 3, 2025

Assurance Reimbursement Management™

Optum is a registered trademark of Optum, Inc. in the U.S. and other jurisdictions. All other brand or product names are the property of their respective owners. Because we are continuously improving our products and services, Optum reserves the right to change specifications without prior notice. Optum is an equal opportunity employer.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Table of contents

Workflow 3
- Real-time request 3
- Batch requests 3
- Viewing responses 3

Setup changes 3
- Enrollments 4
- Eligibility profile configuration 4
- Automation 4
- Group functions 5

Eligibility edit errors & code sets 5
- Eligibility edit errors 5
- Suggested test cycle 6
- When it just doesn't work 6
  - Common mistakes 6
  - Restoring a bridge routine file 7
- Bridge routine reference 7
  - Valid syntax 7
  - Valid commands 7
  - Valid operators 8
  - /SELECT, /OMIT, and /SET_OCC functions 10
  - /VAR functions 14
  - /FUNC functions 15
- Sample bridge routines 26
  - Add data to a field 26
  - Throw away a claim 26
  - Change data based on data in another field 26
  - Strip out unneeded data by LOB 27
  - Flag a claim for review 27
  - Combine service lines 27
  - Set field to blank based on Editmaster requirements 27
  - Multiple bridge routines to get a job done 28

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Workflow

The Eligibility Edits platform supports the same workflow in MDE as using the retired Medicare common working file (CWF). The advantage of Eligibility Edits is that you can send eligibility requests at any time instead of waiting for your FI's open system window.

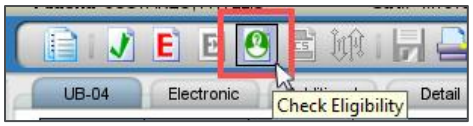


With Medicare Direct Entry (MDE), Assurance or a user checks patient eligibility and then runs front-end edits. The user corrects or overrides errors. Then, Assurance or the user can re-check eligibility and front-end edits as needed before sending MDE claims directly to the payer's system.

While the standard workflow remains unchanged with the switch to the Eligibility Edits platform, there are some differences in how a user performs some tasks (e.g., batch eligibility request). The platform also supports added functionality for MDE claims. Users can now send a real-time eligibility request to Medicare directly from the claim itself.

Real-time request

To submit a real-time eligibility request while viewing an MDE claim, the user clicks the Check Eligibility button in the toolbar.

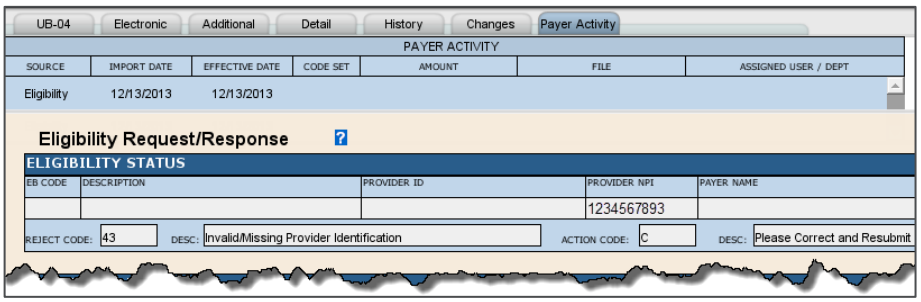


Batch requests

Users can submit a batch of eligibility requests for MDE claims by navigating to the Check Eligibility filter (**Claims** > **Eligibility** > **Check Eligibility**).

Viewing responses

In addition to any errors placed on the MDE claim, a user can review the 270/271 eligibility request and response transactions in the **Payer Activity** tab.



Setup changes

Optum manages your required enrollments for electronic 270/271 transactions with MDE payers. The Eligibility Edits platform automatically checks eligibility for all MDE claims during claim submission (bridging).

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Enrollments

Enrollments are required to use the Eligibility Edits platform to send and receive the 270/271 eligibility transactions for MDE claims. To add another line of business outside of MDE claims, customers should contact their account executive to contract for the full Eligibility Edits add-on module.

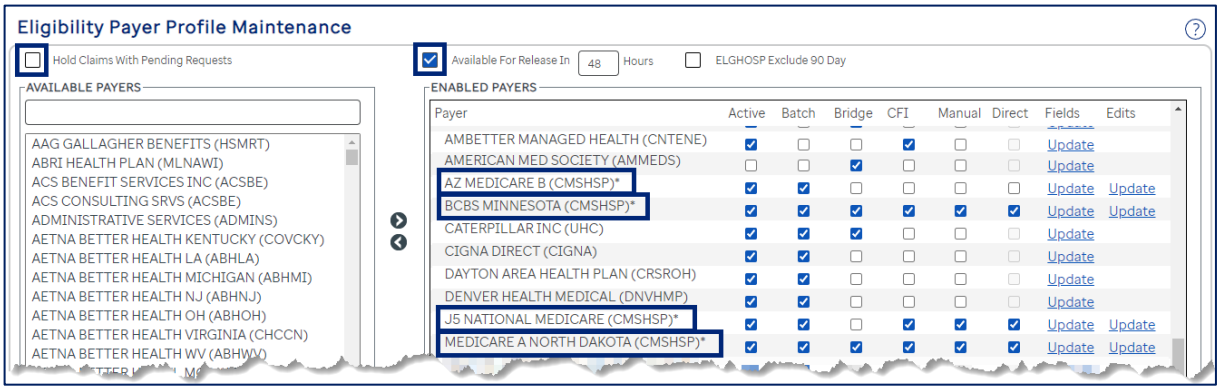
Eligibility profile configuration

Path: Maintenance > Master Setup > Eligibility Profile

The Eligibility Edits feature uses a master system file called the Eligibility Profile to set up lines of business (LOBs). The Assurance Implementation team sets up and activates the MDE LOBs for the customer, including the automation of eligibility checks during bridging. Customers only need to change the master file if they do not want the default settings.

The screenshot below shows a sample profile.

- Medicare LOBs include the **CMSHSP** label.
- By default, Assurance releases all accepted claims to payers even if the claim has a pending eligibility request. To prevent this scenario, the “Hold claims with pending requests” and “Available for release in 48 hours” checkboxes are activated for MDE customers. Assurance then holds accepted claims until it receives and attaches an eligibility response unless there is no response within 48 hours. When the set period elapses, Assurance allows the release of pending claims. The pending status remains on the claims. Customers can change the allowed elapsed amount of time in the eligibility profile.



Automation

Path: Maintenance > Auto Process > Auto List

Assurance Eligibility Profile can be configured to automatically checks eligibility for MDE claims during bridging. This set up replaces most MDE eligibility auto list functionality. This is a typical sequence of automated events:

1. Claims bridge into Assurance. Assurance runs Eligibility Edits.
2. The DE Check Edits job runs, either triggered by completion of bridging or set to run on a schedule.
3. Assurance runs front-end DE reports based on a scheduled job.

Assurance also provides the ability to check Eligibility Edits for Medicare Direct claims with auto modules. Using the auto modules, you can define a claim filter to select the specific claims that request Eligibility Edits responses.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Group functions

Path: Maintenance > Master Setup > Group

Several system functions are associated with Eligibility Edits. These functions should be assigned to users as appropriate.

Function name	Location	Group	Description
Eligibility: Cancel Request	Claims > Eligibility > Cancel Pending Requests	Billing	The user can cancel pending eligibility requests so that new requests can be sent.
Eligibility: Edit Profile	Maintenance > Master Setup > Eligibility Profile	Admin	The user can configure and modify payers to process eligibility requests.
Eligibility: Manual Request	Claims > Eligibility > Manual Eligibility Check	Billing: ADV	The user can manually keystroke and submit eligibility requests.
Eligibility: Override Errors	Checkbox in error text message	Billing: ADV	In the error text message for the errored field, the user can check the override checkbox to manually override an eligibility error.
Eligibility: Print Eligibility	1. Reports > Create Reports 2. Print Claim on toolbar	Billing	The user can print single and batch claims with Eligibility requests and responses.
Eligibility: Submit Request	1. Claims > Eligibility > Check Eligibility 2. Check Eligibility button on toolbar	Billing	The user can create a batch eligibility request using the claim filter or a single eligibility request while viewing the claim.
Eligibility: View Profile	Maintenance > Master Setup > Eligibility Profile	Admin	The user can view profiles of payers set up to process eligibility requests.
View Payer Activity	Payer Activity tab	Billing	The user needs this function to be able to view the 270/271 eligibility request and response transactions. Please note this function also allows users to review any associated 277 claim status and 835 remit transactions.

Eligibility edit errors & code sets

The Eligibility Edits platform offers parity with the MDE eligibility edits from the CWF. Eligibility Edits also offers more code sets and a special Eligibility claim filter you can use for reporting and follow-up purposes.

Eligibility edit errors

Review the table for a list of the available errors and error messages within the Eligibility Edits platform. Most of the error names match the MDE eligibility error names. The exception is the ELIGCHK error. This error flags insufficient or invalid data on the claim to submit a valid 270 eligibility request. Once a user reviews and addresses these claim errors, the user can resubmit an eligibility request directly from the claim.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Suggested test cycle

- Establish your test baseline. Before you check in the changed bridge routine file, submit the test file. Note which claims are accepted and which contain errors. Write down the number of errors for each claim.
- Review the bridge routine file for other routines that use the field you are changing. Be sure these routines still work correctly.
- Check in the changed bridge routine file and submit the test file again. Use View/Modify Claims to verify if your routine is working as expected.
- Verify that your bridge routine does not affect any claims it shouldn't. Your submission file **must contain** a variety of claims.
- When a bridge routine does not work as expected, check the bridge log. Bridge logging records many bridge routine errors.
- When developing a set of routines to do a job, test after each routine to see if the process is working the way you intended. Nothing is worse than spending hours writing multiple routines that must work together only to have a typo buried in the third routine. You must troubleshoot all the routines, not just one.
- Throughout testing, submit the test file, check its claims, and delete the claims. This reduces the possibility that editing will work the test claims to acceptance and possibly release them. (If you need to see the claims later, you can undelete them.)
- Check to ensure that the other routines that use the field you are manipulating still work.
- Don’t panic. If your routine is not working or if you can’t figure out how to write something, put the original bridge routine file back in place and take a break.

When it just doesn’t work

Common mistakes

- Typos - Use cut and paste of commands and field names to reduce typos.
- Misused syntax - If there doesn't appear to be any typos, check Section 0 for syntax and usage rules.
- Not testing - **Big mistake!** You will only do this once.

The bridge log will show you most of these errors. However, if you’ve done everything you can think of, the bridge log shows no errors, and your routine still does not work, try these steps:

- Ask someone else to look at your routine. Sometimes a second set of eyes can find something you can’t.
- Check to see if another routine earlier or later in the file is affecting your new routine. Search by field name.
- If your routine changes data, run a Claim Changes History Report on the test claims to see if any data changes occurred.
- Comment out everything in the bridge routine file except your new routine and submit your claims.
- If the new routine doesn't work as the only routine in the file, the problem is in the new routine itself. You may need to comment out individual lines and test.
- When the routine works, restore the rest of the routines, testing between each restoration. (See why a fail-safe version of the bridge routine file is a must?)

Customer Support is familiar with the syntax of bridge routines and may be able to answer basic questions for you. Specific answers to your questions may be billable at the standard published support rates for your facility.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Restoring a bridge routine file

If you have checked in a changed bridge routine file and need to take a break, restore the original bridge routine file so claims can be submitted. This is another place where your copy of the original is required.

1. Rename the file on which you are working.
2. Rename the original back to its original name.
3. Check in the original.

When you are ready to start again:

1. Check out the original file and rename it to safe name.
2. Rename your work file to the correct original name.

Bridge routine reference

Custom bridge routines are specific to a single site. They are named for the file in which they are stored: the B$<key>.307 file. Bridge routines are written in a simple custom scripting language. The bridge routine scripting language is, in turn, read and processed by the bridge process.

Bridge routines are designed to customize the standard bridge processing to close the gaps between what your information system can produce and what is required by Assurance and the payer. The goal of bridge routines is to reduce the amount of manual labor needed to bill patient claims as cleanly and efficiently as possible.

This section references the commands and syntax of the scripting language used to write bridge routines. It contains all the available commands, functions, and operators. As new commands are added to the bridge routine scripting, they will be added to this section.

Valid syntax

- Comment lines in a bridge routine begin with a #.
- Executable lines begin with a forward slash (/).
- Bridge routines use the abbreviated field names. You can find this information by opening a claim and clicking into the field. A line shows in the lower left corner of the browser window. Example displays:

  Static field: Patient_LastName (10282, P_LNAME)

  This line contains the expanded field name (Patient_LastName), the field ID (10282), and the abbreviated field name (P_LNAME).

  Recurring Field: Payer_Name 1 (10422, TP_PAYER)

  This line contains the expanded field name (Payer_Name), the index value for the recurrence (1), field ID (10422), and the abbreviated field name (TP_PAYER).

Valid commands

Commands begin a bridge routine line.

Command	Detail

/BANK=	**Designates the beginning of a list of values to be entered into a "bank" for later use.** "Banks" must be defined prior to using them. You cannot exceed 25 lines per bank, or 256 characters per line. You can define up to 300 different bank names in a .307 file. If you need more than 300, you can re-use the same bank name(s) as many times as you want.
	`/BANK=PROC_CODES`
	`0870,0886,0887,0889,1311,132,1319,133,1341-1343,1351,1359,1361-1363,1369`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Command	Detail

/FUNC=	**Function (command) to run.** (Options and arguments used with /FUNC are defined later in this section.)
	`/FUNC=SET(RELTRANS,N)`
	`/FUNC=ADDERR(RELTRANS,RELTRANSQ)`

/LABEL=	**Designates the beginning of a new bridge routine.** Label is usually a number. It must be unique in the file. `/LABEL=1`

/MSG=	**Designates the beginning of an error message to be defined for later use by the ADDERR function.** Messages must be defined prior to using them. They must be less than 70 characters per line, and no more than 25 lines per message. Pre-defining messages with /MSG is only necessary for multiple-line messages since a single-line message can be passed directly into the ADDERR function. You cannot define more than 25 different message names in a 307 file; however, you can re-use the same message name(s) multiple times.
	Example: Names and defines the message:
	`/MSG=RELTRANSQ`
	Review service line charges for medical necessity compliance and make adjustments to diagnosis when approved by physician.
	*Another routine sets **RELTRANS** to **Q** and places that message on the RELTRANS field:*
	`/LABEL=23`
	`/FUNC=SET(RELTRANS,Q)`
	`/FUNC=ADDERR(RELTRANS,RELTRANSQ)`

/OMIT=	**Do not continue running the bridge routine if the condition exists.** (Opposite of /SELECT=) `/OMIT=LOB(MEDICARE)`

/SELECT=	**Selection criteria** - continue running the command if the condition exists.
	`/SELECT=TOB(OUTPATIENT)`
	`/SELECT=COMPARE(PRNPCODE,INBANK,PROC_CODES,4)`
	`/SELECT=ANY(SL_RCODE,EQ,100-219,3)`

/SET_OCC=	**Determine which occurrences of a reoccurring data record are to be set** (Uses "ALL" or "ANY"). `/SET_OCC=ALL(SL_RCODE,EQ,310,3)`
/VAR1= **/VAR2=** **/VAR3=**	**Temporary storage variables.**

Valid operators

Operators act on field data. Some operators examine data; others change data. Operators are used with many commands and options. Many operators in this table have more than one recognized form. In these cases, the recognized forms are separated by commas: EQ, = . When writing a bridge routine, you may use EQ or =:

```
/SELECT=ANY(SL_RCODE,EQ,100-219,3)
/SELECT=ANY(SL_RCODE,=,100-219,3)
```

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Operator	Detail
ADD, PLUS, +	Add values of named fields

CONTAINS	String contains. Do not specify wildcards, lists, or ranges. Examples:
	`/SELECT=compare(P_CNTRL,CONTAINS,"999")`
	Continue only if "999" appears anywhere within the patient control number.
	`/SET_OCC=all(SL_DESC,CONTAINS,"LAB")`
	Select all charges with "LAB" appearing anywhere within the description.

DATE_EQ	Date equal to
DATE_GE	Date greater than or equal to
DATE_GT	Date greater than
DATE_LE	Date less than or equal to
DATE_LT	Date less than
DIVIDE, DIVIDED BY, /	Divide first value by second value. `/FUNC=CALC(OTHAMTPD,DIVIDE,100,OTHAMTPD)`
EQ, =	Equal to
GE, >=	Greater than or equal to
GT, >	Greater than
INBANK	Is value in the named bank?
[IndexName]	Refers to the appropriate index value of the named reoccurring field. This value is usually a number. Examples: `SL_RATE[0]` is the first occurrence `SL_RATE[1]` is the second occurrence For Payer fields, you can use the actual **IndexName**. Examples: `PRIMARY` means Index 0 `SECONDARY` means Index 1 `TERTIARY` means Index 2 `ACTIVE` means current payer index (based on PAYERIND) The index value displays as part of the field information line in the lower left of the browser window when hovering over a claim field. Example display: `ServiceLine_Date 3 (23052, SL_DATE)` In this example, the index value is 3. It is the third occurrence of the field. When translating the displayed index value to `IndexName`, subtract one from the displayed value. The `IndexName` value for this occurrence is 2.
INRANGE	Is value in the named range (i.e., 100- 219)?
LE, <=	Less than or equal to
LT, <	Less than

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Operator	Detail
MULTIPLY, TIMES, *	Multiply values
NE, !=	Not equal to
NUM_EQ	Number equal to
NUM_GE	Number greater than or equal to
NUM LE	Number less than or equal to
STR_EQ	String equal to
SUBTRACT, -, MINUS	Subtract first value from second value `/FUNC=CALC(TOT_CHRG,-,C_VAMT,TP_DUEA)`

/SELECT, /OMIT, and /SET_OCC functions

Options in this table can be used with /SELECT, /OMIT, and /SET_OCC commands, except as noted. (Examples use /SELECT.)

These listed money fields support a value of 0.00 as the dollar amount:

- C_VAMT
- TOT_AMT
- SL_TOT
- P_AMTPD
- P_ESTA
- TP_PATRESPAMT
- TP_DUEA
- TP_PRIA
- T_AMTPD
- SL_CHG
- SL_PATLIABAMT
- SLA_PAYAMT
- SLB_PAYAMT

The recommended practice for writing routines is to use the new BLANK_0 keyword. Use BLANK_0 with functions to check for entries that are blank, 0, or 0.00. The pre-existing BLANK keyword now only targets a true empty value.

Function	Detail
ALL	**Are all occurrences of field equal to given value?** **Syntax:** ALL(field name, operator, value, length) 1. field name - valid field ID 2. operator - see list of valid operators in previous section 3. value - value to compare 4. length - length of comparison **Example:** `/SET_OCC=all(tp_plan,eq,100-101,3)`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	Notice that ranges use a dash.
ANY	**Is any occurrence of field equal to given value?** **Syntax:** ANY(field name, operator, value, length) 1. field name - valid field ID 2. operator - see list of valid operators in previous section 3. value - value to compare 4. length - length of comparison **Example:** `/SELECT=ANY(SL_RCODE,EQ,300,3)`

ANYDIAG	**Is any diagnosis code equal to given value?**
	With 5.3, this has been updated to read BANKs. ANYDIAG/ANYPROC will continue if any item in the bank matches. With 7.2, this function no longer requires the length specified as the last argument.
	Syntax: ANYDIAG(value, length) or ANYDIAG(INBANK, bankname)
	1. Value – value or range of diagnosis codes to compare
	2. Length – number of places to compare [optional with 7.2]
	Examples: `/SELECT=anydiag(E,1)`
	`/SELECT=ANYDIAG("V761,V7610-V7612,V7619",4)`
	Note: Looks at principal, other, and admitting codes. The list of values uses quotes to group the parameter values together.
	`/LABEL=TEST`
	`/SELECT=ANYDIAG(INBANK,TESTBANK)`
	`/FUNC=SET(HOLDCODE,TEST)`

ANYDUPS	**Do occurrences of field(s) match the given criteria?**
	Usage: Use to identify duplicates or other special matching criteria, for reoccurring records such as charge lines, value codes, etc. This may be used with /SELECT to look for any match, or with /SET_OCC to perform an action on the matching occurrences.
	Syntax: ANYDUPS(field, operator, value, field, operator, value, …)
	1. Field: valid field ID
	2. Operator - see list of valid operators in this guide, or use special operators:
	a. `MATCH`: field values must be identical for both occurrences, and also match the specified value (or use with value "?" to find any duplicate field values)
	b. `ONEEQ`: field value for only one occurrence must match the specified value (both occurrences cannot match the value)
	c. Other operators (`EQ`, `INBANK`, `CONTAINS`, etc.): field value for both occurrences must match the specified value.
	3. Value - value or range to compare
	Example 1: # Check if duplicate value codes exist
	`/SELECT=ANYDUPS(c_vcode,match,?)`
	Example 2: # Matching rev codes (219-999), hcpcs, and date will get flagged with a custom error
	`/SET_OCC=ANYDUPS(sl_rcode,match,219-999,sl_hcpcs,`
	`match,?,sl_date,match,?)`
	`/FUNC=ADDERR( sl_rcode,"DUPLICATE CHARGE")`
	Example 3: # Rev code does not have to match, as long as it is in the range 300-310. Hcpcs and date must match.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail

	`/SELECT=ANYDUPS(sl_rcode,eq,300-310,sl_hcpcs,match, 80000-89999,sl_date,match,?)`
	Example 4: # Look for a hcpcs Q0081 that has the same date as a hcpcs 90780-90799 (does not find duplicate Q0081 or 90780-90799)
	`/SELECT=ANYDUPS(sl_hcpcs,oneeq,Q0081,sl_hcpcs,oneeq, 90780-90799,sl_date,match,?)`

ANYPROC	**Is any UB procedure code equal to given value?**
	With 5.3, this has been updated to read BANKs. ANYDIAG/ANYPROC will continue if any item in the bank matches. With 7.2, this function no longer requires the length specified as the last argument.
	Syntax: ANYPROC(value,length) or ANYPROC(INBANK,bankname)
	1. value - value or range to compare
	2. length - length of comparison [optional with 7.2 release]
	Examples:
	`/SELECT=ANYPROC(9900-9999,4)`
	`/LABEL=TEST`
	`/SELECT=ANYPROC(INBANK,TESTBANK)`
	`/FUNC=SET(HOLDCODE,TEST)`

COMPARE	**Used to compare contents of named file with value named in the arguments.**
	Usage: Use on static or first occurrence of a recurring data field. *Do not use with /SET_OCC*
	Syntax: COMPARE(field name, operator, value, length)
	1. field name - valid field ID
	2. operator - see list of valid operators above
	3. value - any value or field ID, including BANK
	4. length - length of comparison
	Examples:
	`/SELECT=COMPARE(MCAID_NO,STR_EQ,H70090455,9)`
	`/SELECT=COMPARE(C_VCODE,INRANGE,01- 10,2)`
	`/SELECT=COMPARE(PRNPCODE,INBANK,PROC_BANK,5)`
	`/SELECT=COMPARE(STM_FDAT,DATE_EQ,STM_TDAT)`

DUPKEY	**Compare parameters with previous claim to determine if this is a duplicate claim.** Syntax: DUPKEY(field name, ..., ...) Parameters: List field ID's that make a unique claim key Example 1: `/SELECT=DUPKEY(p_cntrl)` Example 2: `/SELECT=DUPKEY(p_cntrl,p_lname,p_fname,p_mi)`
EMAST	**Electronic Carrier master used on this claim.** Usage: *Do not use with /SET_OCC* Syntax: EMAST(carrier, carrier, carrier, ...) Arguments: Any valid electronic carrier master Example: `/SELECT=EMAST(HE_X000)`
EMPTY	**Is field empty?** Usage: *Do not use with /SET_OCC*

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	Syntax: EMPTY(field name, field name, ...) Arguments: Any valid field name Example: `/SELECT=EMPTY(PRNPCODE,OTH1PCOD,OTH2PCOD )`

FIND	**Is any occurrence of field equal to given value?**
	Syntax: FIND (field name, operator, value, length)
	1. field name - valid field ID
	2. operator - see list of valid operators above
	3. value - value to compare
	4. length - length of comparison
	Example: `/SELECT=find(sl_rcode,eq,300,3)`
	Note: Same as /SELECT=any, except allows a reoccurring field to be assigned to a static field:
	`/SELECT=FIND(c_occ,eq,A1,2)`
	`/FUNC=SET(p_bday,c_ocdat)`
	The above does not work using /SELECT=any because the first index of c_ocdat is assumed when assigning to p_bday, whereas /SELECT=find causes the appropriate occurrence code index to be used.

LOB	**Line of Business to which this command applies.** Usage: *Do not use with /SET_OCC* Syntax: LOB(line of business, line of business, ...) Arguments: medicare, mcare, mc, medicaid, mcaid, md, bcbs, blue, bc, champus, champ, commercial, comm, any other payer (aetna, prudential, ...) Example: `/SELECT=LOB(MEDICARE,MEDICAID)`
PAPER	**Is this a paper claim?** Usage: *Do not use with /SET_OCC* Syntax: PAPER() Arguments: None Example: `/SELECT=PAPER()`
REFAE	**Get the Referring Physician name (L, F, M).** Note: *REF_ID must be present on the claim.* Syntax: REFAE() Arguments: None Example: `/SELECT=REFAE()`
SAMEDAY	**Are the two dates the same?** Usage: *Do not use with /SET_OCC* Syntax: SAMEDAY(field name, field name). 1. field name – valid date field 2. field name - valid date field Example: `/SELECT=SAMEDAY(STM_FDAT,STM_TDAT)`
SUMM_GE	**Is the sum of all occurrences of the field greater than or equal to the named value?** Usage: Use on recurring numeric data fields. *Do not use with /SET_OCC.* Syntax: SUM_GE(field to sum, range field, range, value)

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	1. field to sum - any reoccurring numeric field 2. range field - field to check against range 3. range - range of values to check 4. value - value to check sum of fields against Example: `/SELECT=SUMM_GE(SL_TOT,SL_RCODE,100-219,3000.00)`
SUMM_LE	**Is the sum of all occurrences of the field less than or equal to the given value?** Usage: Use on reoccurring numeric data fields. *Do not use with /SET_OCC.* Syntax: SUM_LE(field to sum, range field, range, value) 1. field to sum - any reoccurring numeric field 2. range field - field to check against range 3. range - range of values to check 4. value - value to check sum of fields against Example: `/SELECT=SUMM_LE(SL_TOT,SL_RCODE,100-219,3000.00)`
TOB	**Type of Bill to which this command applies.** Usage: *Do not use with /SET_OCC.* Syntax: TOB(type of bill, type of bill, ...) Arguments: inpatient, inp, outpatient, out, 131, 11?, etc. Example: `/SELECT=TOB(13?,83?)`

/VAR functions

Options in this table can be used with any /VAR1, /VAR2, and /VAR3 command as well as with /SELECT, /OMIT, and /SET_OCC commands.

Function	Detail
add	**Returns the sum of two numbers** Usage: Can be used with any field whose value is a number Syntax: add(value1, value2) 1. value1 – field that contains a number or any number 2. value1 – field that contains a number or any number Example: `/VAR1=add( P_COVD, P_NCD )`
days	**Returns the number of days between two dates** Usage: UB only. Do not use with /SET_OCC Syntax: days(field name, field name) 1. field name - valid date field 2. field name - valid date field Example: `/VAR1=days(STM_FDAT,STM_TDAT)`
divide	**Returns the quotient of two numbers** Usage: Can be used with any field whose value is a number Syntax: divide(value1, value2) 1. value1 – field that contains a number or any number 2. value1 – field that contains a number or any number Example: `/VAR1=divide(TOT_AMT,2)`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
multiply	**Returns the product of two numbers** Usage: Can be used with any field whose value is a number Syntax: multiply(value1, value2) 1. value1 – field that contains a number or any number 2. value1 – field that contains a number or any number Example: `/VAR1=multiply(SL_RATE,SL_UNIT)`
subtract	**Returns the difference between two numbers** Usage: Can be used with any field whose value is a number Syntax: subtract value1, value2) 1. value1 – field that contains a number or any number 2. value1 – field that contains a number or any number Example: `/VAR1=subtract(P_COVD,P_NCD)`
SUMM	**Add up all occurrences of a numeric field matching the given criteria.** Syntax: summ(field to sum, range field, range, length) 1. field to sum - any reoccurring numeric field 2. range field - field to check against range 3. range - value or range of values to check 4. length - length of comparison Example: `/VAR1=summ(SL_TOT,SL_RCODE,"270,271",3)`
tally	**Returns the number of occurrences matching the value passed in.** Syntax: tally(field name, range, length) 1. field name - valid date field ID 2. range - value or range to compare against 3. length - length of comparison Example: `/VAR1=tally(sl_rcode,"270,271",3)`

/FUNC functions

Options in this table can be used with the /FUNC command. With the 7.2 release, functions such as RECALC and ROLL_SL place 0.00 in the target fields when appropriate.

Function	Detail

ADDDAYS	**Add the given number of days to a date.**
	Syntax: ADDDAYS(field, days)
	1. field - field ID to be modified, must contain data in a recognizable date format
	2. days - number of days to add, use negative number to subtract days
	Example 1:
	`/FUNC=SET(remarks7,stm_fdat)`
	`/FUNC=ADDDAYS(remarks7,-3)`
	`/SET_OCC=ALL(sl_date,date_lt,remarks7[0])`
	Example 2:
	`/FUNC=SET(remarks8,stm_tdat)`
	`/FUNC=ADDDAYS(remarks8,3)`
	`/SET_OCC=ALL(sl_date,date_gt,remarks8[0])`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail

ADDERR	**Add an error message to the specified field.**
	Syntax: ADDERR(field, msg name)
	For the 5.3 release, this function has been updated to read the field lists saved by the new ALLPROC and ALLDIAG functions (example 3).
	1. field - valid field ID
	2. msg name – message or name of message previously defined with /MSG
	Examples:
	1) `/FUNC=ADDERR(P_CNTRL, "REVIEW PCN"`
	(add short message directly)
	2) `/FUNC=ADDERR(RELTANS,RELTRANSQ)`
	(add named message created with the /MSG function)
	3) To add an error to every non-blank diag code that isn't "ABCDE":
	`/LABEL=TEST2`
	`/OMIT=ALLDIAG("ABCDE")`
	`/FUNC=ADDERR(FIELDLIST,"TEST2")`
	Notes:
	o If the message contains spaces, enclose it in quotes.
	o Adding a "Y" as a third parameter will trigger an error that will be removed by validation in CFI. For example, `/FUNC=ADDERR(TYP_BIL, "REVIEW TYPE OF BILL",Y)` creates an error that is automatically removed by validation. `/FUNC=ADDERR(P_CNTRL, "REVIEW PCN")` creates an error that must be removed manually with the ***Accept*** button.

ADDREC	**Add a new reoccurring data line.** Usage: Use on reoccurring data fields only Syntax: Addrec(field name, value, ..., ...) 1. field name - valid field ID 2. value - value to set the field with Example: `/FUNC=ADDREC(C_VCODE,50,C_VAMT,120.00)`

ALLDIAG	**Compares the value against all diagnosis codes, and saves a list of matching field IDs (FIELDLIST) and the corresponding POA (FIELDLISTPOA).**
	Usage: If used with SELECT, this function returns the matches. If used with OMIT, this function returns fields that did not match, excluding blank fields. Length is optional and behaves like other functions. Only stops the label if the final field list is empty.
	ALLDIAG only works with SELECT and OMIT. Only SET and ADDERR can use the field lists.
	BANKS are allowed. The function compares each diagnosis code to each bank item. Empties FIELDLIST and FIELDLISTPOA each time this is called.
	FIELDLIST is shared with ALLPROC. Each will overwrite each other.
	Syntax : ALLDIAG(value, length) or (INBANK, bankname)
	1. value
	2. length
	Example: The following example will add an error to every non-blank diag code that isn't "ABCDE".
	`/LABEL=TEST2`
	`/OMIT=ALLDIAG("ABCDE")`
	`/FUNC=ADDERR(FIELDLIST,"TEST2")`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail

ALLPROC	**Compares the value against all proc codes, and saves a list of matching field IDs (FIELDLIST) and corresponding dates (FIELDLISTDATE).**
	Usage: If used with SELECT, this function returns the matches. If used with OMIT, this function returns fields that did not match, excluding blank fields. Length is optional, and behaves like other functions. Only stops the label if the final field list is empty.
	ALLPROC only works with SELECT and OMIT. Only SET and ADDERR can use the field lists.
	BANKS are allowed. The function compares each proc code to each bank item. Empties FIELDLIST and FIELDLISTDATE each time this is called.
	FIELDLIST is shared with ALLDIAG. Each will overwrite each other.
	Syntax : ALLPROC(value, length) or (INBANK, bankname)
	1. value
	2. length
	Example: Following example will change each procedure code with the value “ABCXY”, and its associated date, to “TEST1”.
	`/LABEL=TEST1`
	`/SELECT=ALLPROC("ABCXY")`
	`/FUNC=SET(FIELDLIST,"TEST1")`
	`/FUNC=SET(FIELDLISTDATE,"TEST1")`

APPEND_ID	**Append argument 2 to the value of argument 1, with an optional separator.**
	Usage: Can be used in conjunction with SET_OCC
	Syntax: APPEND_ID(field name, value, separator)
	1. field name - valid field ID
	2. value - what to append to field
	3. separator (optional) - what to place in between the field and value
	Example 1: `/FUNC=append_id(mcare_no,X)`
	Example 2: `/FUNC=set(box38_4,pay_city)`
	`/FUNC=append_id(box38_4,pay_st,",")`
	`/FUNC=append_id(box38_4,pay_zip," ")`

APPEND_SEQ	**Append the current claim sequence number for the download file to the given field, and pad zero out to n number of digits.** Usage: Can be used in conjunction with SET_OCC Syntax: APPEND_SEQ(field name, value) 1. field name - valid field ID 2. value - what to append to field Example: `/FUNC=APPEND_SEQ(BOX2A,4)`
CALC	**Calculate a value for two fields.** Syntax: CALC(field1, operation, field2, dest field) 1. field1 - first field ID in operation 2. operation - math sign 3. field2 - second field ID in operation 4. dest field - field ID to assign result Example: (Calculates covered charges and places that value into the associated sl_tot) `/FUNC=CALC(SL_TOT,-,SL_NCC,SL_TOT)`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail

COMB_DUPS	**Combine duplicate Service Lines where the following fields match:**
	UB - Rev Code, HCPCS, Modifiers, Serv Date
	1500 - HCPCS, Modifiers, Serv From Date, Serv Thru Date;
	matching on service dates or drug fields is optional.
	Syntax: COMB_DUPS() or
	COMB_DUPS(values) or
	COMB_DUPS(range field, values, flag, flag, flag)
	Arguments: **range** - Range of Revenue Codes for UB, or HCPCS for 1500, to which this command applies. (default is All) or range field - service line field to compare against
	values - range of values to check
	flag1 - (Y/N) whether to match service dates (default is N)
	flag2 - (Y/N) whether to combine units (default is Y)
	flag3 - (Y/N) whether to include three drug fields in evaluation (default is N). Fields are SL_NDC, SL DRUGMEASCD, and SL_DRUGMEASQTY
	Examples: `/FUNC=COMB_DUPS()`
	`/FUNC=COMB_DUPS("220- 999")`
	`/FUNC=COMB_DUPS(SL_HCPCS,"10000-69999",N)`
	`/FUNC=COMB_DUPS(SL_PROC,"80000-89999",Y,N)`
	`/FUNC=COMB_DUPS(SL_HCPCS,"80000-89999",Y,Y,Y)`

COMBINE_SL	**Combine Service Lines that match the criteria.**
	We recommend that Optum build combine_sl bridge routines.
	Syntax: Combine_Sl(old, new, desc, hcpcs, flag, flag)
	1. old - Revenue Code range to combine
	2. new - Revenue Code to combine into
	3. desc - default service line description
	4. hcpcs - HCPCS range to combine
	5. flag - Flag whether to total combined HCPCS
	6. flag - Flag whether different dates should be separated.
	Notes: Setting flag 5 to Y will replace the HCPCS code on the new service line with 8000n (n= number of lines combined). Setting the flag to N will retain the HCPCS code of the first new service line.
	Setting flag 6 to Y will combine only service lines that use the same date. If there are multiple dates for the given revenue codes, then multiple service lines will be created, with each service line having a unique date.
	Example: `/SET_OCC=ALL(SL_HCPCS,INBANK,CPT_CODES)`
	`/FUNC=SET(SL_HCPCS,"PANEL")`
	`/FUNC=COMBINE_SL(300,300,"LAB CHEM GROUP","PANEL",Y,Y)`
	(Resulting service line will be revenue code 300, HCPCS will be 8000n, and there will be a separate service line for each different date.)

CUT	**Cut characters from the left/right of a field.** Usage: Use on specific data field and index or use SET_OCC first to determine the occurrences of the field to cut Syntax: cut(field name, mode, num) 1. field name - valid field ID

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	2. mode LTRIM = cut from left, RTRIM = cut from right 3. num - number of characters to cut Example 1: `/FUNC=CUT(TP_CERT[1],LTRIM,2)` Example 2: `/FUNC=CUT(TP_PAYER,LTRIM,10 )` Example 3: `/SET_OCC=ALL(TP_PAYER,INRANGE,1-9,1)` `/FUNC=CUT(TP_PAYER,LTRIM,10)`
DATALOOKUP	**Attempts to use <filename> based on the <inifilename> to lookup the key in the file and assign any additional data to the claim. (similar to TEXTIMPORT)** Syntax: DATALOOKUP(<filename>,<inifilename>,<key>) 1. filename - Identifier of the additional file to use. Do not give the path as the file must live in the client's sites directory with an extension of TBL. 2. inifilename - Name of the INI file to use for parsing extra data. Do not give the path as the file must live in the client's sites directory. 3. key – Field to use as key to find in file. Example: `DATALOOKUP(DESC_FILE,RCODE_LOOKUP.INI,SL_RCODE[999])` Note 1: Can specify index of 999, like SL_RCODE[999] or TP_PAYER[999] to let the code loop over all occurrences, via Count OCC.
DELDIAG	**Delete diagnosis codes and remove any gaps.** With 5.3, this has been updated to read BANKs. DELPROC/DELDIAG will delete each field listed in the bank. Syntax: DELDIAG(value, length) or DELDIAG(INBANK, bankname) 1. value - value or range of codes to delete 2. length - length to compare Example: `/FUNC=DELDIAG(E,1)` Note 1: Does not affect UB admitting diagnosis code Note 2: For 1500 the descriptions are also deleted and shifted with their corresponding codes
DELDUPS	**Delete duplicate reoccurring data lines that meet the parameters.** Usage: Use on reoccurring data fields only Syntax: DELDUPS(field name, value, length) 1. field name - valid field ID 2. value - value to compare field with Example: /FUNC=DELDUPS(C_COND,20) /FUNC=DELDUPS(C_COND,ALL)
DELOCC	**Delete a reoccurring data record.** Usage: First occurrence of a record is always 0 (not 1) Syntax: DELOCC(record header, occurrence) 1. record header - tprty1 third party one record (payer) - tprty2 third party two record (employer) - tprty3 third party three record - value value code record - srvline service line record - cond condition code record

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	- occrnce occurrence code record - occspan occurrence span code record 2. occurrence - which occurrence or record Example: `/FUNC=DELOCC(TPRTY1,1) ...DELETE SECONDARY` `/FUNC=DELOCC(TPRTY1,2) ...AND TERTIARY PAYERS`
DELPROC	**Delete UB procedure codes and remove any gaps.** With 5.3, this has been updated to read BANKs. DELPROC/DELDIAG will delete each field listed in the bank. Syntax: DELPROC(value, length) or DELPROC(INBANK, bankname) 1. value - value or range of codes to delete 2. length - length to compare Examples: `/FUNC=DELPROC(9900-9999,4)` Note: Procedure dates are also deleted and shifted along with their corresponding codes.
DELREC	**Delete a recurring data line.** Usage: Use on reoccurring data fields only Syntax: DELREC(field name, value, length) 1. field name - valid field ID 2. value - value to compare field with 3. length - length of field to compare Example: `/FUNC=DELREC(C_VCODE,50,2)` - or - Usage: Use in conjunction with SET_OCC Syntax: Delrec(field name) Arguments: field name - valid field ID Example: `/SET_OCC=ALL(TP_PAYER,EQ,UMWA,4)` `/FUNC=DELREC(TP_PAYER)`
DELSAME	**Delete payer information on this payer line if it is the same as primary.** Syntax: DELSAME() Arguments: None Example: `/SELECT=LOB(MEDICARE)` `/FUNC=DELSAME()`
DISCARD	**Discard a claim.** Syntax: DISCARD() Arguments: None Example: `/FUNC=DISCARD()`
DISCARD_XOVER	**Discard a claim with the ability to know that it was discarded because it was crossed over.** Syntax: DISCARD_XOVER() Arguments: None Example: `/FUNC=DISCARD_XOVER()`
FMT_ID	**Format Physician Id number (right justify, zero fill).**

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	Usage: Use on static data, primary occurrences, reoccurring fields, and specific field indexes. Syntax: FMT_ID(field name, length) 1. field name - valid field ID 2. length - number of digits in ID Example: `/FUNC=FMT_ID( APHYSID, 6 )` (TN123 will become TN000123)
FMT_UPIN	**Format a physician name into LAST<space>FIRST<space>MI.** Max sizes: LAST, 8; FIRST, 3; MI, 1 Notes: 1) Assumes name is in LAST FIRST MI format, where any part can be of any length 2) Works on static data or primary occurrences Example: `/FUNC=FMT_UPIN(ATTPHYS)`
FMT_UPIN_16	**Format a physician name to LAST<space>FIRST<space>MI.** Max Sizes: LAST,16; FIRST, 8; MI 1 Notes: 1) Assumes name is in LAST FIRST MI format, where any part can be of any length 2) Works on static data or primary occurrences Example: `/FUNC=FMT_UPIN_16(ATTPHYS)`
KILLPREV	**Delete the first of duplicate claims (instead of last).** Usage: Use in conjunction with DUPKEY Syntax: KILLPREV() Arguments: None Example: `DUPKEY(P_CNTRL)` `/FUNC=KILLPREV()`

MATCH_SET	Usage: Use this function to match two fields or sets of fields, and if they match, then set the <target field> to the value of the <source field>. If <source field> is empty/blank, and <optional field> is specified, use <optional field> instead.
	Arguments:
	1. <target field> field to set with the data from the source or optional field.
	2. <source field> field to use to populate the field to set if the matches occur.
	3. <match A field 1> this is the field to compare against <match B field 1>
	4. <match B field 1> this is the field to compare against <match A field 1>
	5. <match A field 2> this is the field to compare against <match B field 2> (optional but must have a corresponding <match B field 2>)
	6. <match B field 2> this is the field to compare against <match A field 2> (optional but must have a corresponding <match A field 2>)
	7. <optional field> use this field to populate <target field> if the <source field> is empty
	Examples:
	`/FUNC=MATCH_SET(SL_PROCDESC,DTL_DESC,SL_HCPCS,DTL_HCPCS, SL_DATE,DTL_DATE,SL_DESC)`
	if SL_HCPCS = DTL_HCPCS and SL_DATE = DTL_DATE, then SL_PROCDESC will contain the data in DTL_DESC
	if DTL_DESC is blank/empty, it will be populated SL_DESC (if that data exists, otherwise, it won't be touched)
	`/FUNC=MATCH_SET(SL_PROCDESC,DTL_DESC,SL_HCPCS,DTL_HCPCS, SL_DATE,DTL_DATE)`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail

	if SL_HCPCS = DTL_HCPCS and SL_DATE = DTL_DATE, SL_PROCDESC will contain the data in DTL_DESC
	if DTL_DESC is blank/empty, it will be won't be touched, i.e. no default field was given.
	`/FUNC=MATCH_SET(SL_PROCDESC,DTL_DESC,SL_HCPCS,SL_DESC)`
	if SL_HCPCS = DTL_HCPCS, SL_PRODDESC will contain the data in DTL_DESC (note: no second comparison)
	if DTL_DESC is blank/empty, it will be populated SL_DESC (if that data exists, otherwise, it won't be touched).
	`/FUNC=MATCH_SET(SL_PROCDESC,DTL_DESC,SL_HCPCS,DTL_HCPCS)`
	if SL_HCPCS = DTL_HCPCS and SL_DATE = DTL_DATE, SL_PRODDESC will contain the data in DTL_DESC
	if DTL_DESC is blank/empty, it will be won't be touched, i.e. no default field was given.

PAD_FLD	**Insert data in a field from after the given position for a specified number of characters.** Syntax: PAD_FLD(field, pos, len, value) 1. field - field name to be padded 2. pos – number of positions to skip before beginning the pad 3. len - pad this many characters 4. value - character of the pad a. BLANK - Insert spaces b. field - Insert this field's data c. string - Insert this character string **Note:** For recurring fields, PAD_FLD only works with SELECT=ANY. It does not support SET_OCC or a recurring field with a specified index. Example 1: (To pad P_LNAME field from position one for one-character length) `/FUNC=PAD_FLD(P_LNAME,1,1,')` OBRIAN would change to O'BRIAN
PARSE_COD	**Parse the diagnosis code (or procedure code) from a code description including both the code and the description.** Syntax: PARSE_COD(code id, desc id) 1. code id - field ID for code 2. desc id - field ID for description Example: `/FUNC=PARSE_COD(PRNPCODE,PRNPDESC)`
PARSE_DELIM	**Split one record into several based on a delimiter.** Usage: Can be used in conjunction with SET_OCC Syntax: PARSE_DELIM(field name, field name, value, value) 1. field name where results go 2. field name containing data 3. delimiter position 4. delimiter Example: `/FUNC=PARSE_DELIM(GF_1B,REMARKS8,1,"|")`
PARSE_RATE	**Parse the last 6 digits of sl_desc into sl_rate.** Syntax: PARSE_RATE()

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	Arguments: None Example: `/FUNC=PARSE_RATE()`
PREFIX_ID	**Prefix parameter 1 with the value of parameter 2.** Usage: Can be used in conjunction with SET_OCC Syntax: PREFIX_ID(field name, value) 1. field name - valid field ID 2. value - what to prefix field with Example: `/FUNC=PREFIX_ID(MCARE_NO,X)`
RECALC	**Forces a recalculation of the 001 total and the non-covered charges based on current service lines.** Usage: UB only. Syntax: RECALC() Arguments: None Example: `/FUNC=RECALC()`

ROLL_SL	**Combine Service Lines that match the criteria.**
	Usage: This is almost the same as combine_sl but keeps the data from the first matching service line found.
	Syntax: ROLL_SL(old rev, new rev, old hcpcs, new hcpcs, flag, flag, flag)
	1. old rev - Revenue Code range to combine
	a. (ignored for 1500 forms)
	2. new rev - Revenue Code range to combine into
	a. (ignored for 1500 forms; first one found is used,
	b. if not found then nothing is rolled)
	3. old hcpcs - HCPCS range to combine
	4. new hcpcs - HCPCS range to combine into (first one
	a. found will be used, if not found then nothing is rolled)
	5. flag - (Y/N) whether to combine units (default is Y)
	6. flag - (Y/N) whether to combine only if service dates
	a. are equal (default is N)
	7. flag - (Y/N) whether to leave old HCPCS present with
	a. zero amount (default is N)
	Note: With the 7.2 release, a flag of Y will place 0.00 in SL_TOT (Inst) or SL_CHG (Prof).
	1500 Example:
	`/FUNC=ROLL_SL(?,?,72193,74160)`
	UB Examples:
	`/FUNC=ROLL_SL(300-319,300-319,80000-80019,84443,Y,Y)`
	`/FUNC=ROLL_SL(???,???,10000,10000-69999,N,N,Y)`
	`/FUNC=ROLL_SL(450,450,?,10000-69999)`
	`/FUNC=ROLL_SL(762,450,?????,?????,N)`
	Note 1: "?" includes a blank value.
	Note 2: “???” includes any three-character value, “????” includes any four-character value, etc.
	Note 3: When matching on service dates, multiple service dates are combined.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
SEP_HCPCS	**Separate SL_HCPCS and SL_M1 from SL_RATE and blank out SL_RATE.** Usage: Use in conjunction with SET_OCC Syntax: SEP_HCPCS() Arguments: None Example: `/SET_OCC=ALL(SL_RCODE,EQ,300-999,3)` `/FUNC=SEP_HCPCS()`

SET	**Set a field to a value.**
	Usage: Use on static or primary occurrence data fields or use SET_OCC first to determine the occurrences of the reoccurring record to set. For the 5.3 release, this function has been updated to read the field lists saved by the new ALLPROC and ALLDIAG functions (example 3).
	Syntax: SET(field name, value, ..., ...)
	1. field name - valid field ID
	2. value - value to set the field with
	Example 1: `/FUNC=SET(RELTRANS,N)`
	Example 2: `/SET_OCC=ALL(SL_RCODE,EQ,310,3)`
	`/FUNC=SET(SL_RCODE,300)`
	Example 3: Change each procedure code with the value “ABCXY”, and its associated date, to “TEST1”.
	`/LABEL=TEST1`
	`/SELECT=ALLPROC("ABCXY")`
	`/FUNC=SET(FIELDLIST,"TEST1")`
	`/FUNC=SET(FIELDLISTDATE,"TEST1")`

SETSTATEREPORTINGONLY	**Flag a claim as state-only. This routine applies to the original State Data Reporting feature. It does not apply to claims when the customer uses the Regulatory Reporting Tool (RRT) to submit state claims.** Claims will never be sent to any payer – they do not even require a payer. State-only claims are in a claim state by themselves so that they do not mingle with standard payer claims. The most common reason for setting a claim to state-only is use of the State Data Reporting feature. (The RRT feature uses a separate system code to mark claims that should not go to payers, so this routine does not apply.) Usage: Flag a claim as a state-only claim Syntax: SETSTATEREPORTINGONLY() Arguments: None Example: `/SELECT=COMPARE(TP_PAYER,EQ,"SELF PAY")` `/FUNC=SETSTATEREPORTINGONLY()`
SPLITCLM	**Split the claim into two or more claims.** Syntax 1: SPLITCLM(MAXCHGS,X) Desc: Creates a new claim for each x service lines. 1) "maxchgs" 2) x - num of service lines to appear on each claim not including the 001 service line. Example: `/FUNC=splitclm(MAXCHGS,56)` A claim with 60 service lines will be split into 2 claims. The first contains 56 charges; the second has 4.

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
	Syntax 2: splitclm(DAY) splitclm(MONTH) splitclm(YEAR) Desc: Creates new claim(s), distributing the service lines so each claim contains service lines for only one day (or month or year). Service lines which have no service dates are left on the original claim. Parameters: "day" or "month" or "year" Example: `/FUNC=splitclm(MONTH)` Syntax 3: splitclm(CHGS, service line field, <values>) Desc: Each service line which matches the specified field will be removed from the original claim and moved onto a second claim. 1. "CHGS" 2. service line field (SL_RCODE, SL_PROC, etc) 3. <values> is list or range of values to compare. "?" is a wild card, so 54? matches 540,541...549. Ranges can be included by using a dash. For example, 545-549. Banks cannot be used here. Example: `/FUNC=splitclm(CHGS,SL_RCODE,401,403)` results in the original claim (without the 401 and 403 service lines) and a new UB containing the 401 and 403 service lines. Example: `/FUNC=splitclm(CHGS,SL_PROC,51234,78???-81000)` results in the original HCFA claim (without the services lines with the specified HCPCS), and a new HCFA claim which contains those service lines. Syntax 4: splitclm(UB92HCFA, field, <values>) Desc: Each service line which matches the specified service line field will be removed from the UB and placed on a new HCFA-1500 claim. UBs with more than 48 service lines will not be split. Split Maintenance on the Product Support site contains the mapping of UB fields to HCFA fields. 1. "UB92HCFA" 2. field (SL_RCODE, SL_HCPCS, P_CNTRL, etc) 3. <values> is list or range of values to compare. "?" is a wild card, so 54? matches 540,541...549. Ranges can be included by using a dash. For example, 545-549. Banks cannot be used here. Example: `/FUNC=splitclm(UB92HCFA,SL_RCODE,540)` results in the original UB (without the 540 service line) and a new professional claim which contains the 540 service line. Syntax 5: splitclm(HCFAUB92,field,<values>) Desc: Each service line which matches the specified service line field will be removed from the HCFA and placed on a new UB claim. Split Maintenance on the Product Support site contains the mapping of HCFA fields to UB fields. 1. "HCFAUB92" 2. field (SL_PROC, P_CNTRL, etc) a. <values> is list or range of values to compare. "?" is a wild card, so 54? matches 540,541...549. Ranges can be included by using a dash. For example, 545-549. Banks cannot be used here Example: `/FUNC=splitclm(HCFAUB92,SL_RCODE,540)` results in the original UB (without the 540 service line) and a new professional claim which contains the 540 service line. Syntax 6: splitclm(DUPLICATE) Desc: Creates an exact copy of the claim Example: `/FUNC=splitclm(DUPLICATE)`

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Function	Detail
STMTDATE	**Modify the stmt from and thru dates to match the service dates. Will ignore blank service line dates.** Example: `/FUNC=STMTDATE()`
STRIP	**Strip specified titles from last name. This will remove the characters from the end of the string ONLY to avoid removing characters that should have been left in.** Example: `/FUNC=STRIP(P_LAST,MR,MRS,JR,SR)`
TEXTIMPORT	**Attempts to import <filename> into NDC_BRIDGETEXTIMPORT based on the inifilename. Will then lookup the key in the object for addition of additional data.** Syntax: TEXTIMPORT(<filename>,<inifilename>,<key>) 1. <filename> this is the name of the additional file to import. Do not give the path. Full filename and path will be generated based off the bridge filename. 2. <inifilename> this is the name of the INI file to use for parsing extra data. Do not give the path, file must live in the client's sites directory. 3. <key> field to use as key to find in file - most likely based off P_CNTRL. *McKesson Proprietary* submission format uses a double key: P_CNTRL and C_DATE Example: `/FUNC=TEXTIMPORT(BCTWCEP,SERIES.INI,P_CNTRL)`
UNPROCESS	**Send a claim to the unprocessed file.** Syntax: UNPROCESS() Example: `/FUNC=UNPROCESS()`

Sample bridge routines

This section contains a sampling of bridge routines with interpretations to help familiarize you with the syntax and range of possibilities for bridge routine activities.

Add data to a field

Purpose: For Medicare claims, set the physician qual code to 1G

```
/LABEL=5
   /SELECT=COMPARE(TP_PAYER,EQ,"MEDICARE",8)
   /FUNC=SET(TP_PHYQC,1G)
```

Throw away a claim

Purpose: Do not process blank claims into the database.

```
/LABEL=0
   /SELECT=COMPARE(P_CNTRL,EQ,BLANK)
   /FUNC=DISCARD()
```

Change data based on data in another field

Purpose: Set insured's sex if relationship is 02 (spouse). Two routines are required--one for F, one for M.

```
/LABEL=1
   /SELECT=COMPARE(P_SEX,EQ,M)
   /SET_OCC=ALL(TP_REL,EQ,02)
   /FUNC=SET(TP_SEX,F)
```

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

```
/LABEL=2
   /SELECT=COMPARE(P_SEX,EQ,F)
   /SET_OCC=ALL(TP_REL,EQ,02)
   /FUNC=SET(TP_SEX,M)
```

Strip out unneeded data by LOB

Purpose: On Medicare and Medicaid claims, strip the service lines if the first three characters of the Revenue Code are in the 960-980 range and recalculate service lines.

```
/LABEL=5
   /SELECT=LOB(MEDICARE,MEDICAID)
   /SET_OCC=ALL(SL_RCODE,INRANGE,960-989,3)
   /FUNC=DELREC()
   /FUNC=RECALC()
```

Flag a claim for review

Purpose: Name and set the bank of revenue codes to 331, 332, and 333. If the claim is Medicare outpatient and the revenue code is in the bank, set RELTRANS to "C" and add "CHEMOTHERAPY REVIEW" to F/L 2B. This routine ensures that the claim cannot go to the payer unless someone looks at the claim and validates it.

```
/BANK=BANK3 331,332,335
/LABEL=20
   /SELECT=LOB(MEDICARE)
   /SELECT=TOB(OUTPATIENT)
   /SELECT=ANY(SL_RCODE,INBANK,BANK3,3)
   /FUNC=SET(RELTRANS,C)
   /FUNC=SET(BOX2B,"CHEMOTHERAPY REVIEW")
```

Combine service lines

Purpose: On CHAMPUS claims, combine service lines with revenue codes 450 and 459. Add "EMERGENCY ROOM" as the description. Combine all lines with these revenue codes and the same date into one line.

```
/LABEL=40
   /SELECT=LOB(CHAMPUS)
   /SELECT=TOB(OUTPATIENT)
   /SELECT=ANY(SL_RCODE,EQ,459,3)
   /SELECT=ANY(SL_RCODE,EQ,450,3)
   /FUNC=COMBINE_SL(459,450,"EMERGENCY ROOM","00000-99999",Y,Y)
```

Set field to blank based on Editmaster requirements

Purpose: Blank out Accident Indicator if the claim uses the PE_I000 editmaster. Note that the comment indicates what triggered the need for this bridge was written.

```
# BLANK ACCIDENT INDICATOR FOR MEDICAID OF ILLINOIS (I000). PER EDIT
MASTER REQUIREMENTS.
/LABEL=18B
   /SELECT=EMAST(PE_I000)
   /FUNC=SET(OTHACC,BLANK)
```

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.

Multiple bridge routines to get a job done

Purpose: Set Physician Qualifier Code to 1G if the Physician ID is a UPIN and to OB if the ID is not in UPIN format. Note number of bridge routines required to catch every case. Depending on the number of LOBs, this kind of bridge routine set can be very long!

```
/LABEL=10
   /SELECT=EMAST(HE9XXXX)
   /SELECT=COMPARE(TP_ATTID,NE,"A-Z",1)
   /SELECT=COMPARE(TP_ATTID,NE,??????)
   /FUNC=SET(TP_PHYQC,0B)
/LABEL=10B
   /SELECT=EMAST(HE9XXXX)
   /SELECT=COMPARE(TP_ATTID,EQ,"A-Z",1)
   /SELECT=COMPARE(TP_ATTID,EQ,??????)
   /FUNC=SET(TP_PHYQC,1G)
/LABEL=10C
   /SELECT=EMAST(HE9AAAA)
   /SELECT=COMPARE(TP_ATTID,NE,"GA",2)
   /SELECT=COMPARE(TP_ATTID,NE,????????)
   /FUNC=SET(TP_PHYQC,0B)
/LABEL=10D
   /SELECT=EMAST(HE9AAAA)
   /SELECT=COMPARE(TP_ATTID,EQ,"GA",2)
   /SELECT=COMPARE(TP_ATTID,EQ,????????)
   /FUNC=SET(TP_PHYQC,1G)
/LABEL=10E
   /SELECT=LOB(MEDICARE)
   /SELECT=COMPARE(TP_ATTID,EQ,"A-Z",1)
   /SELECT=COMPARE(TP_ATTID,EQ,??????)
   /SET_OCC=ALL(TP_ATTID,EQ,??????)
   /FUNC=SET(TP_PHYQC,1G)
```

© 2025 Optum, Inc. and/or one of its subsidiaries. All rights reserved.