# TIBCO Object Service Broker

## Quick Reference

*Software Release 6.0*

*July 2012*



## PF Keys

### Supported on Every TIBCO Object Service Broker Screen

| | |
|---|---|
| <PF1> | Help |
| <PF3> | Save changes and exit |
| <PF12> | Cancel changes and exit<br>Note: Works as refresh in an IMS TM environment. |
| <PF24> | Refresh<br>Note: In an IMS TM environment, cancels changes and exits. |

### Operating Where Applicable

| | |
|---|---|
| <PF2> | Documentation screen or view message log |
| <PF7> | Scroll up |
| <PF8> | Scroll down |
| <PF9> | Recall previous primary command |
| <PF10> | Scroll left |
| <PF11> | Scroll right |
| <PF13> | Print |
| <PF22> | Delete |

### Scrolling Commands

| | |
|---|---|
| C | Up or down to cursor position |
| H | Half screen up or down |
| M | Top, bottom, left edge, or right edge |
| n | Number of lines up or down |
| P | One screen in any direction |

## Primary Commands

**APPLY** *object_command*
[WHERE *field relational operator value* {& / | *field relational operator value*}]
[ORDERED
[ASCENDING/DESCENDING] *field* {& / | *field relational operator value*}
{& ORDERED [ASCENDING/DESCENDING] *field* {& / | *field relational operator value*}}]

```
APPLY P WHERE UNIT ='TEST' ORDERED
DESCENDING TIME
```

**FIND** *field relational operator value* {& / | *field relational operator*}
[ORDERED
[ASCENDING/DESCENDING] *field* {& / | *field relational operator value*}
{& ORDERED [ASCENDING/DESCENDING] *field* {& / | *field relational operator value*}}]

```
FIND LOCATION LIKE '*ville'
```

**ORDERED** [ASCENDING/DESCENDING] *field*
{& [ASCENDING/DESCENDING] *field*}

```
ORDERED DEPTNO AND ORDERED DESCENDING LNAME
```

**SELECT** [*field relational operator value* {&/ | *fieldrelationaloperatorvalue*}]
[ORDERED
[ASCENDING/DESCENDING] *field* {& / | *field relational operator value*}
{& ORDERED [ASCENDING/DESCENDING] *field* {& / | *field relational operator value*}}]

```
SELECT DEPTNO=10 ORDERED LNAME
```

## Mask Characters

You can add a display mask to a field of semantic data type B or P from the Report Definer, Report Generator, Screen Definer, and various shareable tools.

### Numeric Mask Characters

| Digit Placeholder | Leading Character | Example |
|---|---|---|
| 9 | 0 | $PIC(12, '999') => '012' |
| Z | blank | $PIC(12, '$ZZZ') => '$ 12' |
| N | null | $PIC(12, '$NNN') => '$12' |
| * | * | $PIC(12, '$***') => '$*12' |

### Other Mask Characters

| Character | Purpose |
|---|---|
| V | Character after V is decimal placeholder. |
| + | Display plus (+) for positive number and dash (-) for negative. |
| - | Display blank for positive number and dash (-) for negative. |

## Keywords

| | | |
|---|---|---|
| AND | ASCENDING | BROWSE |
| CALL | COMMIT | DELETE |
| DESCENDING | DISPLAY | END |
| EXECUTE | FORALL | GET |
| IN | INSERT | LIKE |
| LOCAL | NOT | NULL |
| ON | OR | ORDERED |
| PRINT | REPLACE | RETURN |
| ROLLBACK | SCHEDULE | SIGNAL |
| TO | TRANSFER-CALL | UNTIL |
| UPDATE | WHERE | |

## Syntax

| Syntax | Valid Lengths (bytes) |
|---|---|
| B<br>Binary | 2 – 12 |
| C<br>Fixed Length Character String | • 1 – 127 (key field or parameter)<br>• 1 to the maximum length for an occurrence<br>• 1 – 254 (display fields) |
| F<br>Floating Point | • 4, 8, and 16<br>• 24, 56, and 112 binary digits of precision for the mantissa<br>• $5.4 \times 10^{-79}$ to $7.2 \times 10^{75}$ for approximate range |
| P<br>Packed Decimal | 1 – 16 (holds 1 – 31 decimal digits) |
| RD<br>Raw Data | 5 to the maximum length for an occurrence |
| UN<br>Unicode | • 2 to the maximum length for an occurrence<br>• Must be even |
| V<br>Variable Length Character String | • 1 – 127 (key field)<br>• 1 to the maximum length for an occurrence<br>• 1 – 254 (display fields) |
| W<br>Double-byte or Single-byte Character String | 4 to the maximum length for an occurrence<br>Valid on z/OS only |

## Wildcard Characters

You can use LIKE to perform partial matches when using SELECT or FIND. The string that you use must be enclosed in single quotation marks. You can use the following wildcard characters in conjunction with LIKE:

| Character | Purpose | Example |
|---|---|---|
| * | A string of zero or more characters. | 'OBJ*' for a string starting with "OBJ". |
| ? | A single character. | 'OBJ?' for any 4-character string starting with "OBJ". |

## Semantic Data Types

| Semantic Data Type | Description | Possible Operators | Valid Syntax |
|---|---|---|---|
| C<br>Count | Integers for integral count (no fractions) | Relational<br>Concatenation<br>Assignment<br>Arithmetic | B, C, P, UN, V |
| D<br>Date | Used for dates; include at least a year | Relational<br>Concatenation<br>Assignment<br>Arithmetic<br>(+ or -) | B |
| I<br>Identifier | Unique Identifier | Relational<br>Concatenation<br>Assignment | B, C, P, UN, V |
| L<br>Logical | Information meeting a Yes or No condition | Relational<br>Concatenation<br>Assignment<br>Logical | C |
| Q<br>Quantity | Real numbers used for measurement | Relational<br>Concatenation<br>Assignment<br>Arithmetic | B, C, F, P, UN, V |
| S<br>String | Character string data | Relational<br>Concatenation<br>Assignment | C, RD, UN, V, W |
| Typeless | Literals defined in TIBCO Object Service Broker rules | Relational<br>Concatenation<br>Assignment<br>Arithmetic<br>Logical | B, C, F, P, RD, UN, V, W |

## Standard Line Commands

| | |
|---|---|
| **D** | Delete the object. |
| **P** | Print a hardcopy. |
| **S** | Select the object.<br>In the Rule Editor, splits a line. |
| **I** | Insert a new line. |
| **C** | Copy a line. |
| **M** | Move a line to a new position. |
| **A** and **B** | For a move or copy, specify a destination after (**A**) or before (**B**) the line where the command is to be placed. |
| **R** | In the Rule Editor, replicates a line. |

# Operators

A filled square (■) indicates that the operation can be performed; an empty square (❒) indicates that the operation is invalid.

## Relational Operators

### Equality Relational Operators (=, ¬=)

| | Count | Date | Identifier | Logical | Quantity | String | Typeless |
|---|---|---|---|---|---|---|---|
| Count | ■ | ❒ | ■ | ❒ | ❒ | ❒ | ■ |
| Date | ❒ | ■ | ❒ | ❒ | ❒ | ❒ | ■ |
| Identifier | ■ | ❒ | ■ | ❒ | ❒ | ■ | ■ |
| Logical | ❒ | ❒ | ❒ | ■ | ❒ | ❒ | ■ |
| Quantity | ❒ | ❒ | ❒ | ❒ | ■ | ❒ | ■ |
| String | ❒ | ❒ | ■ | ❒ | ❒ | ■ | ■ |
| Typeless | ■ | ■ | ■ | ■ | ■ | ■ | ■ |

### Ordering Relational Operators (<, >, <=, >=)

| | Count | Date | Identifier | Logical | Quantity | String | Typeless |
|---|---|---|---|---|---|---|---|
| Count | ■ | ❒ | ❒ | ❒ | ❒ | ❒ | ■ |
| Date | ❒ | ■ | ❒ | ❒ | ❒ | ❒ | ■ |
| Identifier | ❒ | ❒ | ■ | ❒ | ❒ | ❒ | ■ |
| Logical | ❒ | ❒ | ❒ | ❒ | ❒ | ❒ | ❒ |
| Quantity | ❒ | ❒ | ❒ | ❒ | ■ | ❒ | ■ |
| String | ❒ | ❒ | ❒ | ❒ | ❒ | ■ | ■ |
| Typeless | ■ | ■ | ■ | ❒ | ■ | ■ | ■ |

## Concatenation Operator (||)

The concatenation operator, a double vertical bar (||), is valid between any two semantic data types. If one of the operands has syntax W, the result has semantic data type string and syntax W. In all other cases, the result has semantic data type string and syntax V.

## Arithmetic Operators

### Addition (+)

| | Count | Date | Quantity | Typeless |
|---|---|---|---|---|
| Count | Count | Date (if COUNT is binary) | ❒ | Count |
| Date | Date (if COUNT is binary) | ❒ | ❒ | Date |
| Quantity | ❒ | ❒ | Quantity | Quantity |
| Typeless | Count | Date | Quantity | Typeless |

### Subtraction (-)

| | Count | Date | Quantity | Typeless |
|---|---|---|---|---|
| Count | Count | ❒ | ❒ | Count |
| Date | Date | Count | ❒ | Date |
| Quantity | ❒ | ❒ | Quantity | Quantity |
| Typeless | Count | ❒ | Quantity | Typeless |

### Multiplication/Division/Exponentiation (*, /, **)

| | Count | Quantity | Typeless |
|---|---|---|---|
| Count | Count | Quantity | Count |
| Quantity | Quantity | Quantity | Quantity |
| Typeless | Count | Quantity | Typeless |

## Assignment Operator (=)

| | Count | Date | Identifier | Logical | Quantity | String | Typeless |
|---|---|---|---|---|---|---|---|
| Count | ■ | ❒ | ■ | ❒ | ❒ | ■ | ■ |
| Date | ❒ | ■ | ❒ | ❒ | ❒ | ■ | ■ |
| Identifier | ■ | ❒ | ■ | ❒ | ❒ | ■ | ■ |
| Logical | ❒ | ❒ | ❒ | ■ | ❒ | ❒ | ■ |
| Quantity | ❒ | ❒ | ❒ | ❒ | ■ | ■ | ■ |
| String | ■ | ■ | ■ | ❒ | ■ | ■ | ■ |
| Typeless | ■ | ■ | ■ | ■ | ■ | ■ | ■ |

## Resultant Syntax

| First Operator Syntax | Second Operator Syntax | Result Syntax | Decimal Places or Length of Result for: | | | |
|---|---|---|---|---|---|---|
| | | | + and - | * | / | ** |
| B(L1) | B(L2) | B(4) | | | | |
| B(L1) | P(L2,D2) | P(8,D) | D=D2 | D=D2 | D=DMAX | D=DMAX |
| B(L1) | F(L2) | F(L2) | | | | |
| P(L1,D1) | B(L2) | P(8,D) | D=D1 | D=D1 | D=DMAX | D=DMAX |
| P(L1,D1) | P(L2,D2) | P(8,D) | D=MIN(MAX(D1,D2),DMAX)) | D=MIN(D1+D2,DMAX) | D=DMAX | |
| P(L1,D1) | F(L2) | F(L2) | | | | |
| F(L1) | B(L2) | F(L1) | | | | |
| F(L1) | P(L2,D2) | F(L1) | L=L1 | L=L1 | L=L1 | L=L1 |
| F(L1) | F(L2) | F(L) | L=MIN(L1,L2) | L=MIN(L1,L2) | L=MIN(L1,L2) | L=MIN(L1,L2) |

## Resultant Syntax Table Key

| | |
|---|---|
| D | Number of decimal places in the result |
| DMAX | Maximum number of decimal places that the result can hold without overflowing the integer portion |
| D1 | Number of decimal places in the first operand |
| D2 | Number of decimal places in the second operand |
| L | Length of the result |
| L1 | Length of the first operand |
| L2 | Length of the second operand |
| OP2 | Value of the second operand |

### Logical Operators

| Word | Symbol | Meaning |
|---|---|---|
| AND | & | • Expression evaluates to true (Y), only if both logical operands are true<br>• Valid only in selection criteria |
| OR | \| | • Expression evaluates to true (Y), if either (or both) logical operand(s) is true<br>• Valid only in selection criteria |
| NOT | ¬ | • Expression evaluates to true (Y), only if the logical expression is false<br>• Valid in selection criteria and in the conditions portion of a rule |

## Date Formats

| Format Code | Meaning | Example |
|---|---|---|
| W | Week # (of year) with no leading 0 (1 or 2 digits) | 1 or 25 |
| WW | Two digit week # (of year) | 01 |
| WWW | Abbreviated weekday | Mon |
| WWWW | Full weekday | Monday |
| M | Numeric month, with no leading 0 (1 or 2 digits) | 3 or 10 |
| MM | Numeric month (2 digits) | 02 |
| MMM | Abbreviated month | Mar |
| MMMM | Full month | March |
| D | Day in month, with no leading 0 (1 or 2 digits) | 5 or 14 |
| DD | Day in month (2 digits) | 02 |
| DDD | Day in year (3 digits) | 074 |
| YY | Last two digits in year | 99 |
| YYYY | Full year | 1999 |
| QQ | Two character quarter | 2Q |
| JD | Julian date | 99.074 |
| CC | Two digit century | 19 |

## Statements

For more information, refer to *TIBCO Object Service Broker Programming in Rules*.

**CALL**

*rule* [(*argument*{, *argument*})] /
[WHERE *arg_name=arg_value* {& *arg_name=arg_value*}];

**COMMIT;**

**DELETE**

*table* [(*parm_value*{, *parm_value*})]
[WHERE *primary_key = expression*] ;

**DISPLAY**

*screen*;

**DISPLAY & TRANSFERCALL**

*display screen* [& TRANSFERCALL
[IN BROWSE/UPDATE]
*rule* [(*argument*{, *argument*})]]/
[WHERE *arg_name=arg_value*{&
*arg_name=arg_value*}];

**EXECUTE**

[IN BROWSE/UPDATE]
*rule* [(*argument*{, *argument*})] /
[WHERE *arg_name=arg_value* {&
*arg_name=arg_value*}];

**FORALL**

*table* [(*parm_value*{, *parm_value*})]
[WHERE [¬] *field / parm_name relational_operator /*
LIKE *expression* { & / | [¬] *field relational_operator /*
LIKE *expression*}]
ORDERED
[ASCENDING/DESCENDING] *field* {& ORDERED
[ASCENDING/DESCENDING] *field*}]
[UNTIL *exception_name* [*table*] {| *exception_name*
[*table*]}] :
*statement*;
{*statement*;}
END;

**GET**

*table* [(*parm_value*{, *parm_value*})]
[WHERE [¬] *field / parm_name relational_operator /*
LIKE *expression* {& / | [¬] *field relational_operator /*
LIKE *expression*}]
[ORDERED

[ASCENDING/DESCENDING] *field* {& ORDERED [ASCENDING/DESCENDING] *field*}]
[WITH MINLOCK];

**INSERT**
*table* [(*parm_value*{, *parm_value*})] /
[WHERE *parm_name=parm_value*
{& *parm_name=parm_value*}];

**ON**
*exception_name[table]:*
*statement;*
*{statement;}*

**PRINT**
*report;*

**REPLACE**
*table* [(*parm_value*{, *parm_value*})] /
[WHERE *parm_name=parm_value*
{& *parm_name=parm_value*}];

**RETURN**
(*expression*);

**ROLLBACK;**

**SIGNAL** *exception_name;*

**SCHEDULE**
[IN BROWSE/UPDATE]
[TO *queue*]
*rule* [(*argument*{, *argument*})] /
[WHERE *arg_name=arg_value* {&
*arg_name=arg_value*}];

**TRANSFERCALL**
[IN BROWSE/UPDATE]
*rule* [(*argument*{, *argument*})] /
[WHERE *arg_name=arg_value* {&
*arg_name=arg_value*}];

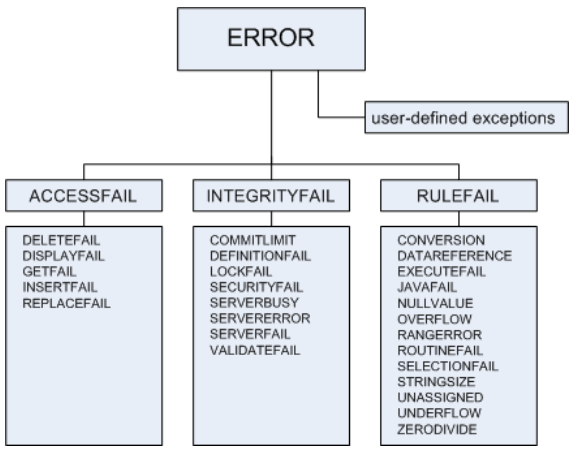
**UNTIL**
UNTIL *exception_name* {| *exception_name*}:
*statement*;
{*statement*;}
END;

**UNTIL...DISPLAY**
UNTIL *exception_name* {| *exception_name*} [DISPLAY {*screen*}]:
*statement*;
{*statement*;}
END;

## Exception Hierarchy

The following diagram illustrates all system exception names in their relative position in the exception hierarchy. Each identification represents a lower level in the hierarchy.

A handler for an exception traps the exceptions that are listed beneath it (i.e., lower in the hierarchy). For example, the ERROR exception traps all detectable errors and the INTEGRITYFAIL exception traps only the exceptions listed from COMMITLIMIT to VALIDATEFAIL.




### Exceptions

The conditions that trigger each of these exceptions are described below in alphabetical order. For more information, refer to *TIBCO Object Service Broker Programming in Rules*.

**ACCESSFAIL**
A table access error is detected or a rule running in browse mode is attempting to update a table.

**COMMITLIMIT**
The limit on the number of updates between synchronization points has been reached.

**CONVERSION**
A value has invalid syntax or cannot be converted to the target syntax.

**DATAREFERENCE**
An error is detected in selection criteria.

**DEFINITIONFAIL**
An error is detected in the definition of a table.

**DELETEFAIL**
The primary key for a DELETE statement does not exist or a rule running in browse mode is attempting to update a table.

**DISPLAYFAIL**
An error is detected when trying to display a screen.

**ERROR**
An error is detected.

**EXECUTEFAIL**
An error is detected in the child transaction.

**GETFAIL**
No occurrence satisfies the selection criteria.

**INSERTFAIL**
The primary key provided for an INSERT statement already exists or a rule running in browse mode is attempting to update a table.

**INTEGRITYFAIL**
Attempt to violate data integrity detected.

**JAVAFAIL**
A Java exception is raised by a called Java external routine.

**LOCKFAIL**
Another transaction is accessing this data in a way that prevents you from accessing the data.

**NULLVALUE**
An arithmetic operator is being applied to a numeric null or a numeric null is being passed as an argument value when a numeric value is required.

**OVERFLOW**
A value is too large to be assigned to the target syntax. The maximum value for the syntax is inserted into the receiving field. All tables are limited to the defined dictionary length. As well, screen and report tables are also limited to the display length.

**RANGERROR**
An argument to a given OSB routine is out of the allowable range.

**REPLACEFAIL**
A primary key provided for a REPLACE statement does not exist or a rule running in browse mode is attempting to update a table.

**ROUTINEFAIL**
A call to an OSB routine cannot complete successfully and a more specific system exception cannot be signaled.

**RULEFAIL**
An error results from incorrect rules coding, given that the dictionary definition of the database is correct.

### SECURITYFAIL
Permission for the requested action is denied.

### SELECTIONFAIL
An error is detected in a table occurrence during the evaluation of selection criteria.

### SERVERBUSY
The requested external database server is not available to process the transaction.

### SERVERBUSY
The requested external database server is not available to process the transaction.

### SERVERERROR
External database server error detected.

### SERVERFAIL
The connection to an external database server broke during a transaction or the external database server failed.

### STRINGSIZE
The receiving string field is too short to contain the full length of the string value being assigned to it. The value is truncated to the length of the receiving field and inserted into that field.

### UNASSIGNED
Reference is being made to a field of a table not assigned a value.

### UNDERFLOW
A value is too small to be represented in the target syntax (usually exponent errors). The minimum value for the syntax is inserted into the receiving field.

### VALIDATEFAIL
An attempt is being made to update a screen or table with invalid data. For example:

- An attempt is being made to insert data into a table that failed a reference check or a non-Y value is being returned from a validation rule.
- Invalid data is being inserted into a screen table from a rule (that is, the data failed the screen table reference check).
- Invalid data existed on the screen when the user left the screen by using the Validation Exit key.

### ZERODIVIDE
Attempt to divide by zero detected.

## External Syntax

| | External Syntax | Import, Export | VSAM | MAP |
|---|---|---|---|---|
| A | Alphabetic (Uppercase). | ✔ | ✔ | ✔ |
| B | Binary, signed (half word or full word). | ✔ | ✔ | ✔ |
| C | Fixed length character string. | ✔ | ✔ | ✔ |
| E | Little-endian binary, signed. | ✔[a] | ✘ | ✔ |
| F | Floating point (short, long, or extended). | ✔ | ✔ | ✘ |
| F | 370 Floating point. | ✘ | ✘ | ✔ |
| G | Packed, neutral (X'OF') sign when positive. | ✔ | ✔ | ✔ |
| H | Hexidecimal. | ✔ | ✔ | ✔ |
| J | ASCII character string. | ✘ | ✘ | ✔ |
| K | Binary, unsigned. | ✔ | ✔ | ✔ |
| M | Numeric (zoned), unsigned. | ✔ | ✔ | ✔ |
| N | Numeric (zoned), signed. | ✔ | ✔ | ✔ |
| NL | Zoned numeric with sign leading | ✔ | ✔ | ✔ |
| NT | Zoned numeric with sign trailing | ✔ | ✔ | ✔ |
| O | Packed, no sign stored (up to 15 digits). | ✔ | ✔ | ✔ |
| P | Packed, signed (up to 31 digits). | ✔ | ✔ | ✔ |
| Q | Quoted character string. | ✔ | ✔ | ✔ |
| R | Little-endian binary, unsigned. | ✔[a] | ✘ | ✔ |
| RD | Raw data | ✔ | ✔ | ✔ |
| T | Text numeric. | ✔ | ✔ | ✔ |
| U | Packed, unsigned (up to 31 digits). | ✔ | ✔ | ✔ |
| UN | Unicode | ✔ | ✔ | ✔ |
| V | Variable character. | ✔ | ✔ | ✔ |
| W | Double-byte and single-byte character string. | ✔[b] | ✔ | ✘ |
| X | Fixed length, mixed case character string. | ✔ | ✔ | ✔ |
| Z | X'00' fill character. | ✔ | ✔ | ✔ |

a. External syntaxes E and R are not valid on z/OS.
b. W is valid for z/OS only.

## Functional Listing of Shareable Tools

### Batch Jobs (z/OS)

BATCH – Submits a batch job to a particular queue, views the status of the batch jobs, and views the queues that are available. (E)

BATCHLOAD_CARDS – Defines input and output to the Batch Load utilities (S6BBRTBL/hrnbrtbl). (E)

$BATCHOPT(batch_option, option_value) – Sets the batch options associated with a SCHEDULE TO statement's batch request, which sends the batch job to a queue. (C)

BATCHUNLD_CARDS(unload_source) – Defines the control cards required by the Batch Unload utilities. (E)

### Data Object Broker Information and Operations

DASTATS – Returns statistical data collected by the Data Object Broker for an individual segment. (F)

DISPLAY_USERS – Displays a list of all users currently logged in to TIBCO Object Service Broker. (E)

HURON_STATS – Displays statistics for performance analysis and problem determination. (E)

OPSTATS – Returns statistical data collected by the Data Object Broker. (F)

S6BNOTIFY – Sends a Notification message to TIBCO Hawk. (C)

S6BTROFF – Terminates tracing initiated by the complementary shareable tool S6BTRON. (C)

S6BTRON – Initiates tracing of rules execution in the current session. (C)

### Dates and Times

$ADD_DATE(date, component, amount) – Adds or subtracts a component (such as a day, week, month, or year) to or from a date and returns a new date value. (F)

$CREATE_DATE(pic_string, date_string) – Converts a string with a specified format to a value of semantic type date. (F)

$DATE_DEFAULT – Returns the default date format used by the installation. (F)

$DATE_LENGTH(pic_string) – Returns the maximum string length of a given date format. (F)

$DATE_PIC(pic_string, date) – Converts a value of semantic type date to a semantic type string. (F)

$DATE_REF(component, duration, date, round) – Adds or subtracts a given number of days to or from a reference date, and converts the number of days returned to units of a day, a week, a month, or a year. (F)

HOUR – Returns the hour of the day when the current transaction started. (F)

LEAPYEAR(year) – Returns a logical value indicating whether a given year is a leap year. (F)

LOCALTIME – Returns the local time when the transaction started. (F)

MINUTE – Returns the minute in the hour the transaction started. (F)

REALTIME – Returns a string containing the current time of day. (F)

$REALTIMER – Returns the number of micro-seconds since 1 January 1980. (F)

SECOND – Returns the second within the minute that the transaction started. (F)

$SLEEP(milliseconds) – Causes the Execution Environment to go dormant. (C)

$SYSTEMDATE – Returns the date when $SYSTEMDATE is called. (F)

TIME – Returns a string containing the time of the day when the transaction started. (F)

$TRXDATE – Returns the start date of the transaction that called this tool. (F)

UTCDATE – Returns the Coordinated Universal Time (UTC) date when UTCDATE is called. (F)

UTCTIME – Returns a string containing the current Coordinated Universal Time (UTC) time. (F)

YEAR – Returns the two-digit year when the transaction started. (F)

## Debugging

DEBUG(rulename) – Invokes the interactive Rule Debugger. (CE)

$GTFSET(function, keyname, userid, termid, all, dob) – Enables or disables the rules tracing facility in the Execution Environment and the Data Object Broker.

@MESSAGEDUMP – Writes traced messages to this table in HEX form, when @TRACEMESSAGES.DUMP is set to Y. (TBL)

@MESSAGETRACE – Stores table access message requests between the Execution Environment and the Data Object Broker collected when using trace facility. (TBL)

@TRACEMESSAGES – Records message traffic between the Execution Environment and the Data Object Broker. (TBL)

## Definitions of Objects

COPY_DEFN(objecttype, instancename, library, environment, srclocation, destlocation, parentonly) – Copies the definition of one or more objects from a source location to a destination location. (C)

COPYDEFN – Copies the definition of one or more TIBCO Object Service Broker objects or object sets. (E)

DEFINE_OBJECTSET(objsetname) – Defines a set of objects or modifies an existing set. (E)

DELETE_DEFN(objecttype, instancename, library, environment, location, parentonly) – Deletes the definition of an object. (C)

DIFF_DEFN(objecttype, instance1, library1, environment1, location1, instance2, library2, environment2, location2, details) – Compares the definitions of two objects and list the differences. (F)

DIFFDEFN – Compares the definitions of one or more pairs of objects and list the differences. (E)

## External Databases and Servers

CHANGE_SERVERID(table_name, old_serverid, new_serverid) – Updates the server ID of any external TIBCO Object Service Broker data types. (E)

@CONFIGURESERVER(type) – Sets and modifies the server configuration parameters for a particular server ID. (E)

DATACOM – Displays a menu to manage the definition of CA-Datacom data. (E)

ESTIMATETBLDFN(num_fields) – Returns an estimate of the maximum CTABLESIZE and XTABLESIZE required for each table type. (E)

$HTTPREQUEST –Returns the HTTP response code from an HTTP request. (F)

IDMS – Displays the main menu used to define a CA-IDMS database to TIBCO Object Service Broker. (E)

IMS – Displays the main menu used to define an IMS/DB database to TIBCO Object Service Broker. (E)

@PEERSERVERID – Directs remote TIBCO Object Service Broker table accesses to a particular peer server on a remote TIBCO Object Service Broker system. (TBL)

RESETXPARM(component, entity, parm name, location) – Resets overrides on server parameters or on default field values set in the Table Definer. (C)

@SERVERERROR(RETURN_MESSAGE) – Returns the last message from the message stack to enable special parsing or handling of the message, which resulted from a request to the TIBCO Object Service Broker external DBMS server. (C)

SETXPARM(component, entity, parm name, value, location) – Overrides a server parameter or the Table Definer default value for a field at table access time. (C)

@STATICSQL – Defines and generates static SQL to be used to access DB2 data. (E)

## External Memory and Routines

$GETENVCOMMAREA(segment#) – Retrieves data passed into TIBCO Object Service Broker from a call-

ing environment that is not TIBCO Object Service Broker. (F)

HLIPREPROCESSOR(hostlang, imbedlang, infile, outfile, listfile, options) – Invokes a language pre-processor to run against COBOL source programs that contain embedded TIBCO Object Service Broker access statements or SQL statements. (C)

@MAP – Registers and allocates storage for use with MAP tables. (TBL)

RETURN_CODE – Returns the return code from the last call of a TIBCO Object Service Broker external routine. (F)

S6BCALL – Invokes a TIBCO-supplied callable routine that requires a specialized environment. (C)

S6BFUNCTION – Invokes a TIBCO supplied function that requires a specialized environment. (F)

$SETENVCOMMAREA(value, segment#) – Passes data from TIBCO Object Service Broker into a calling environment that is not TIBCO Object Service Broker. (F)

$SETSESSIONEND(action, value) – Sets what execution is to take place when a TIBCO Object Service Broker session ends by returning data from the session to an external environment. (C)

### Installation of Components

@INSTALL(component [path]) – Requests the installation of the specified component. (CE)

@UNINSTALL(component) – Requests that the specified component be uninstalled. (CE)

### Load from/Unload to External Files

BATCHLOAD_CARDS – Defines input and output to the Batch Load utilities (S6BBRTBL/hrnbrtbl). (E)

BATCHUNLD_CARDS(unload_source) – Defines the control cards required by the Batch Unload utilities. (E)

EXPOCC_SIZE(table) – Returns the minimum record size required to hold the occurrences of the table being unloaded. (F)

LLOAD(importfile, media) – Loads definitions and data of TIBCO Object Service Broker objects that were previously unloaded from files with names in mixed case or lowercase. (CE)

LOAD(importfile, media) – Loads definitions and data of TIBCO Object Service Broker objects that were previously unloaded. (CE)

LOADER – Loads definitions and data of TIBCO Object Service Broker objects that were previously unloaded, with selection control. (CE)

UNLOAD – Unloads definitions of valid TIBCO Object Service Broker object types from a source system to a z/OS data set or a Windows or Solaris file. Data from table object types could also be unloaded. (E)

UNLOAD_DATA(tablespec, selection, location) – Unloads the data of a table to a z/OS data set or a Windows or Solaris file. (C)

UNLOAD_DEFN(objecttype, objectname, library, location, presentationenv, parentonly) – Unloads the definition of a TIBCO Object Service Broker object to a z/OS data set or a Windows or Solaris file. (C)

UNLOADLIBRARY(library, location) – Unloads all rules in the specified library at the specified location to a z/OS data set or a Windows or Solaris file. (C)

### Mathematical Calculation

ABS(value) – Returns the absolute value of a number. (F)

MAX(x, y) – Returns the larger of two given values. (F)

MIN(x, y) – Returns the smaller of two given values. (F)

MOD(dividend, divisor) – Returns the modulus from dividing the dividend by the divisor. The function MOD handles negative dividends and divisors. (F)

NUM_CHK(val) – Determines if a given string satisfies the TIBCO Object Service Broker definition of a numeric literal. (F)

RANDOM(rangelimit) – Returns a random integer between one and the specified limit. (F)

RANDOMSEED(seed) – Sets the starting seed for the random number generator. (C)

REMAINDER(dividend, divisor) – Returns the remainder from dividing the dividend by the divisor. (F)

ROUND(value) – Returns the specified value rounded to the nearest integer. (F)

### Menus

DEFINE_MENU(menu) – Creates and modifies menus and login screens used within TIBCO Object Service Broker user-defined applications. (E)

DISPLAY_MENU(menuname) – Calls a specific menu into an application. (C)

### Messages and Message Logs

ENDMSG(message) – Sets the transaction completion message. (C)

GETENDMSG – Returns the current value of the end-of-transaction message. (F)

LOG_BROWSE – Displays the contents of the message log. (C)

MESSAGE(utility, msg_num, tokenlist) – Returns a customized message by taking a root message in the MESSAGES table and inserting customizing tokens. (F)

MESSAGE_LOG(msglog, destin) – Preserves the contents of the message log across transactions. (C)

MSGLOG(string) – Inserts the specified string as a line in the TIBCO Object Service Broker message log. (C)

RETURN_SYSMSG – Returns the last $SYSCALL system error message when an exception is raised. RETURN_SYSMSG is a low-level tool that must be called immediately after an exception is trapped. (F)

S6BCALL('api_call',parameters) – Invokes a TIBCO-supplied callable routine that requires a specialized environment. (C)

### Message Oriented Middleware

@MOMCLOSE(connection, queue) – Closes a Message Oriented Middleware (MOM) message queue. (F)

@MOMCOMMIT(connection) – Commits all changes to queues from a single Message Oriented Middleware (MOM) message manager. (C)

@MOMCONNECT(name) – Connects to a Message Oriented Middleware (MOM) message queue (MQ) manager. (F)

@MOMDISCONN(connection) – Disconnects from a Message Oriented Middleware (MOM) message manager. (F)

@MOMGET(connection, queue, table) – Reads a message from a Message Oriented Middleware (MOM) message queue. (C)

@MOMINIT(buflen, mon_type) – Identifies the type of Message Oriented Middleware (MOM) message manager, and initializes its environment (map and control structures) to enable subsequent @MOM calls. (C)

@MOMMAPLENGTH(table_name) – Returns the length of a MAP table. (F)

@MOMOPEN(connection, name) – Opens a Message Oriented Middleware (MOM) message queue. (F)

@MOMOPTION(description) – Queries the numeric equivalent of a Message Oriented Middleware (MOM) option. (F)

@MOMPUT(connection, queue, table, len) – Writes a message to a Message Oriented Middleware (MOM) message queue. (C)

@MOMROLLBACK(connection) – Backs out all database changes from a single Message Oriented Middleware (MOM) message manager since the start of the transaction or since the previous @MOMCOMMIT. (C)

@MOMSETOPT(description) – Sets a MOM option to a specified value. (C)

@MOMSPECIALCMD(manager_name, queue_name, command) – Sends a Message Oriented Middleware (MOM) command to a queue listener task. (C)

@MOMVALIDRC – Checks the return code of a previous command. (C)

@MQSMAP and @MQSMAP_PORT– Registers and allocates storage for use with the MQSMAP table. @MQSMAP is for use on z/OS and @MQSMAP_PORT on Open Systems. (TBL)

### Printing and Output

$BLANKPAGE(titles_yn) – Outputs a blank page. (C)

$FLUSHPRINT – Releases output into the print spool. (C)

$NEWPAGE – Positions subsequent output to the top of a new page. (C)

PRINT_DATA(tablespec, select, sourceloc) – Prints the data of a TIBCO Object Service Broker table. (C)

PRINT_DEFN(object, instance, library, environment, srcloc, parentonly) – Prints the definition of a TIBCO Object Service Broker object. (C)

$PRINTFIELD(string, pos, length, fill, just) – Writes the specified string into the current printline. (C)

$PRINTLINE(text) – Prints a string. (C)

PRINTTABLE(tablespec, pagelength, pagewidth, media) – Prints a table. (C)

PRT_VSCR(vscr, page_length, page_width, page_start, media, mask) – Prints the screen fields of a defined screen in a page format, with or without a mask. (C)

$PUTLINE – Prints the current line constructed by $PRINTFIELD. (C)

$RESETPRINT(length, width, page_number, media) – Resets the output arguments. (C)

$SETP#POS(line_number, left_string, center_string, right_string) – Defines the position and content of page number lines. (C)

$SETPRINT(length, width, page_number, media, clear_title_yn) – Initializes the print attributes or, if they are already set, uses it to clear the titles for the output on the following pages. (C)

$SETTITLE(line_number, left_string, center_string, right_string) – Sets a title or footer to be printed on subsequent pages of output. (C)

$SKIPLINE(count) – Outputs zero or more blank lines. (C)

$TOCPRINT(fill_char) – Prints table of contents. (C)

$TOCPUT(section_name, spacing, numbering_yn) – Puts a line in the table of contents. (C)

S6BCALL – Invokes a TIBCO-supplied callable routine that requires a specialized environment. (C)

S6BFUNCTION – Invokes a TIBCO-supplied function that requires a specialized environment. (C)

### Promotions

ADMIN_RIGHTS – Obtains, releases, or transfers the promotion rights on objects. (E)

DBMAINTLVL – Displays the maintenance level of your TIBCO Object Service Broker database, including any database PTFs applied beyond the maintenance level. (E)

MANAGE_APPLY – Invokes the Promotion facility on the target system. (E)

MANAGE_REQUESTS – Invokes the Promotion facility on the source system. (E)

MANAGE_RIGHTS – Releases or transfers a user's promotion rights on rules, screens, reports, menus, object sets, and tables. (E)

@PROMBINDOBJS – Restores the bind flag settings for the objects updated by @PROMUNBINDOBJS. (E)

PROM_MAIN – Invokes directly the Promotion system. (E)

@PROMUNBINDOBJS – Stores the current setting of the bind flag for a set of objects and reset the values to N in the metadata tables. (E)

RMANAGE_REQUESTS – Manages change requests for systems where the source system is remote to the target system. (E)

### Read from/Write to External Files

ALLOCDSN(ddname, dsname) – Allocates a file to a z/OS DDNAME. (C)

@CLOSEDSN – Closes and frees the current file. (C)

$LISTDSN(dsname_level, buffer_address) – Lists the non-VSAM data sets and Generation Data Group

(GDG) data sets of a certain level, using the z/OS Catalog Search Interface services. (C)

$LISTPDS(pds_name, buffer_address, member_name) – Lists the member names of a partitioned data set (PDS), or retrieves the statistics for a PDS member. (C)

@OPENDSN(dsname) – Specifies the name of the file that is subsequently used by @READDSN or @WRITEDSN. (C)

@READDSN – Returns the next record from the current file. (F)

@WRITEDSN(string) – Writes a record to the current file. (C)

XMLPARSE – Initiates the parsing of an XML document. (C)

XMLSTART(xmldocname, predicate, parm) – Generates an XML document based on the passed data access arguments. (C)

XMLSTARTDSN(outdsn, xmldocname, predicate, parm) – Generates an XML document based on the passed data access arguments and places it in the specified file. (C)

XMLSTARTSETDEST(tablendpec, fieldspec) – Sets up the output table and field for XMLSTART. (C)

XMLSTARTTAB(tablename, format, predicate, parm) – Returns a table to the OIG client. (C)

### Reports

DEFINE_REPORT(reportname) – Defines a new TIBCO Object Service Broker report or modifies an existing one. (E)

GENERATE_REPORT(reportname) – Defines a new report or modifies an existing report using the Report Generator. (E)

$RPTIMMEDIATE(reportname, media) – Sends the records to the output as they are read, without sorting. (C)

$RPTOCCLIMIT(reportname, occlimit) – Limits the number of occurrences used to generate the report. (C)

$RPTOVERLAP(report, reporttable, reportfield, BLANKOVERLAP) – Designates the report tables or report fields that are not to be printed on the overlapping page of a merged report. (C)

$RPTPARMS(reportname, length, width, eject, pagenumber) – Controls explicitly the physical output of a report. (C)

$RPTPRINT(reportname, media) – Prints a report to the media specified. (C)

$RPTSKIPLINES(reportname, reporttable, element, linesbefore, linesafter) – Controls explicitly the spacing of a report. (C)

$SETRPTATTRIBUTE(report, attribute, value) – Sets the attributes of the report that is to be printed. (C)

$SETRPTMEDIUM(report, mediumtype, medium) – Sets the medium to which a report is to be printed. (C)

### Rules and Rules Libraries

$CALLRULE(rulecall) – Invokes a procedural rule. (C)

CHANGERULE – Makes multiple text changes across multiple rules in a library. (CE)

COPYLIB(source_lib, dest_lib) – Copies all the rules from a source library to a destination library. (C)

DEFINE_LIBRARY(libraryname) – Defines a new library, displays a list of the rules in a library, or displays a list of the rules libraries. (E)

EDITRULE(rulename) – Defines a new TIBCO Object Service Broker rule or modifies an existing one. (E)

$EXCEPTION – Returns the name of the last exception signalled within the current transaction by either a SIGNAL statement, a $SIGNAL call, or the system (GETFAIL, ZERODIVIDE, and so on). (F)

$EXCEPTIONOBJECT – Returns the name of the object (for example, a table) associated with the last exception signalled within the current transaction, if that exception is of the type that can be trapped with an ON exception_name object_name: statement. (F)

$FUNCTION(rulecall) – Invokes a functional rule. (C)

INSTALLIB – Returns the name of the currently designated installation library. (F)

LIBID – Returns the name of the currently designated local library. (F)

NOOP – Does nothing. (C)

$RULE_EXISTS(rule) – Checks whether a rule with the given name would be a candidate for execution. The rule can be a rule in the current search path, a TIBCO Object Service Broker routine, or an external routine with an available and executable load module. (F)

$RULENAME(level, transactioncount) – Retrieves the name of a rule from the current execution stack. (F)

RULEPRINTER(rule) – Prints a rule or prints an application structure using the root rule as the base. (E)

SEARCHLIB – Searches all rules or specified rules in a library for a given string. (CE)

$SIGNAL(exception, tablename) – Raises the specified exception. (C)

SYSTEMLIB – Returns the name of the currently designated system library. (F)

$TRXMODE – Retrieves the transaction mode of the current rule. (F)

$TYPECAST – Converts a variable according to the arguments supplied. (F)

### Screens

$BEEP(repetition) – Issues the specified number of beeps from the terminal. (C)

CONFIRMACTION(screen, confirmmsg, key, defaultmsg, table, commandfield) – Issues a confirmation message for a PF key action or for a specified command. (C)

CURSOR_FLDCOL(screen) – Returns the relative column number within the field containing the cursor. (F)

CURSORFIELD(screen) – Returns the name of the field where the cursor is located. (F)

CURSOROCC#(screen) – Returns the occurrence number within the screen table where the cursor is positioned. (F)

CURSOROCC_VALUE(screen, scrtbl, scrfld) – Returns the value of a particular screen field that is selected by the cursor. (F)

CURSORTABLE(screen) – Returns the name of the screen table where the cursor is positioned. (F)

DELETESCREENDATA(screen) – Deletes all the occurrences from all the screen tables of a screen. (C)

DRAW(screenname) – Defines a new TIBCO Object Service Broker screen or modifies an existing one. (E)

ENTERKEY(screen) – Returns the name of last key used when the specified screen appeared. (F)

EVENTSCREEN – Returns the name of the screen that activates the current screen validation rule. (C)

EXIT_DISPLAY – Signals the standard exception EXIT_DISPLAY. (C)

FCNKEY_MSG(screen) – Creates a string containing the function keys defined for a screen. (F)

$GETATTRIBUTE(screen, table, field, attribute) – Queries the current attributes for the field of the screen table, in the specified screen. (F)

$GETCOLOUR(screen, table, field, color_type) – Queries the current color of a screen field. (F)

@PRESENTATIONENV – Returns the name of the presentation environment for the current session. (F)

PROCESS_FCNKEY(screen) – Processes the function keys while a screen is being displayed. (C)

SCREENCOL – Returns the number of columns on the user's physical screen. (F)

SCREENMSG(name, msg) – Displays the given message in the message area of the specified screen. (C)

SCREENROW – Returns the number of rows on the user's physical screen. (F)

$SETATTRIBUTE(screen, table, field, attribute, flag) – Sets attributes for the field of the screen table, in the specified screen. (C)

$SETCOLOUR(screen, table, field, color_type, color) – Sets the color of a screen field. (C)

SETCURSOR(screen, table, field) – Positions the cursor in the field of the screen table, in the specified screen. (C)

SETCURSOR_POS(screen, table, field, occurrence_number, column_offset) – Positions the cursor in the column of the field of the occurrence, in the screen table of the screen. (C)

## Searches for Objects

CROSSREFSEARCH(querystring, querykind, library) – Searches the cross reference index of the specified library to answer a query. (C)

KEYWORDMGR – Ensures that the TIBCO Object Service Broker keyword system conforms to the established formatting standards and that the keyword index table is up-to-date. (E)

KEYWORDSEARCH(querystring, object_type) – Searches the keyword index of a default library to answer a query. (C)

REFMAKER(library) – Rebuilds the global cross reference index. (E)

SEARCH – Searches the keyword or cross reference indexes to answer a query. (E)

SEARCHLIB – Searches all rules or specified rules in a library for a given string. (CE)

## Secondary Indexes

SIXBUILD(table, secondary_key) – Creates a secondary index online for a TDS table. (C)

SIXBUILD_CARDS – Defines the control cards required by the Batch Secondary Index Build utilities. (E)

SIXDELETE(table, secondary_key) – Deletes an existing secondary index. (CE)

## Security

AUDITLOG – Invokes the Query Audit Log tool. (CE)

BATCH_ENABLE(wipe_existing) – Enables all the object sets previously processed using @MAKEMEMBERS. (C)

CREATEUSERS(input_table, modeluser) – Creates a list of new user IDs and adds them to the TIBCO Object Service Broker system. (E)

@MAKEMEMBERS(objectset) – Creates the member list for an object set to be enabled through the BATCH_ENABLE utility. (CE)

@MNG_USERS – Modifies your user security profile. (E)

PURGELOG_BATCH(fromdate, todate, file) – Purges the audit log data from the TIBCO Object Service Broker audit log table and archive to an external file. (CE)

PURGELOG_SCREEN – Specifies the archive file for the audit log data and purges the audit log data to the specified archive file. (CE)

SEC_REBIND(object, parmcat, name) – Rebinds all security data previously bound in the Execution Environment storage. (C)

SECURITY – Invokes the TIBCO Object Service Broker Security Manager main menu. (E)

## Selection Lists

DEFINE_OBJLIST(table) – Defines, for a table, an object list to appear using the Object Manager or modifies an existing object list definition. (E)

OBJECT_MGMT(tablespec) – Displays the contents of a table and enables a predefined set of commands that are unique to the table to operate on the display. (E)

OBJECTMGR(tablespec) – Displays the contents of a table and enables a predefined set of commands that are unique to the table to operate on the display. (C)

OPTIONLISTER(optionlistname) – Displays options in columns and returns the ones selected (C).

SELECT_OBJ(name, type, unit, author, library, location, children, subtype) – Provides a screen that can be used to list and select objects that meet specified criteria. (C)

## Session Options and Parameters

$GETOPT(option_name) – Returns the value of a session parameter or option. (F)

$GETTRANSACTION(name) – Gets a transaction name set by $SETTRANSACTION. (F)

REMOTELOCATION – Returns the current value of the default remote location. (F)

@SESSION – Alters session-related items maintained by this table. (TBL)

@SESSIONCOUNTS – Obtains information on events occurring within the Execution Environment during a session. (TBL)

SESSMGR – Displays the login interface to the Session Manager (workbench).

$SETOPT(parameter, value) – Sets the value of a session parameter or option. (C)

$SETTRANSACTION(field, value) – Returns the current name of a TIBCO Object Service Broker transaction and sets a new name. (F)

SETREMOTELOC(remoteloc) – Sets the default remote location for distributed data processing. (C)

USERID – Returns a string containing the user ID. (F)

### Strings and Text

FROM_UNICODE(unistring, externalcodepage) – Converts a Unicode string to Raw Data encoded in an external Code page.(F)

GEN_TED(tablespec, screenname, screentablename) – Presents a screen where text can be entered and edited under the control of the text editor. (C)

GENBIN(value, length) – Returns a syntax V string containing the same internal binary value as the input numeric value, right-justified.(F)

GENFLOAT(value, length) – Returns a syntax V string containing the same internal float representation as the input float value, left-justified. (F)

GENPACK(value, length, decimal) – Returns a syntax V string containing the same internal packed decimal value as the input syntax P value, right-justified. (F)

$GET_DECIMALS(value) – Retrieves the number of decimal places for an expression. (F)

$GET_MAXSIZE(value) – Retrieves the dictionary size of an expression. (F)

$GET_SIZE(value) – Retrieves the size of an expression. (F)

$GET_SYNTAX(value) – Retrieves the syntax of an expression. (F)

$GET_TYPE(value) – Retrieves the semantic type of an expression. (F)

$GETBINARY(string, offset, length) – Stores character data in binary format. (F)

GETCHAR(string) – Returns the first character from the specified string, removing it from the string. (F)

$GETFLOAT(string, offset, length) – Stores character data in floating format. (F)

$GETPACKED(string, offset, length) – Stores character data in packed decimal format. (F)

HEADSTRING(string, length) – Returns the head portion of the specified string. (F)

LENGTH(string) – Returns the length of the specified string. (F)

LIT_TO_VAL(string) – Converts a string to a typeless field as described in the string. (F)

LOWER_EBCDIC(string) – Converts a string to lowercase EBCDIC characters. (F)

LOWER_UNICODE(string) –Converts a string to lowercase Unicode characters. (F)

LOWERCASE(string) – Converts all uppercase characters in a string to lowercase characters. (F)

MATCH(string, pattern) – Returns the starting position, in characters, of the specified pattern in the specified string, relative to the start of the string. (F)

PAD(string, length, padcharacter, just) – Returns a string padded to a specified length using a pad character, positioning the string to the left, right or center of the padding. (F)

PARSE(grammar_usage, string) – Breaks up an input string into tokens and applies grammar rules to the tokens. (C)

PATTERN_MATCH(string, pattern) – Determines whether a string matches a given pattern. (F)

PEEL(peelchars, string) – Returns the result of removing the specified leading and trailing characters from the specified string. (F)

PEEL_HEAD(char, string) – Removes the specified leading characters from a given string. (F)

PEEL_TAIL(char, string) – Removes the specified trailing characters from a given string. (F)

$PIC(value, mask) – Returns a number in a format specified by a mask. (F)

QUOTE(string) – Returns a string with single quotation marks around it and doubles any single quotation marks in the string. (F)

SCRIPT(source, dest) – Uses commands to format text from a table and store the formatted text in another table. (C)

SEARCH_REPLACE(input_string, replace_this, with_this, else_with_this) – Replaces all occurrences of a pattern with specified characters. (F)

SUBSTRING(string, start, length) – Returns a selected portion of a string. (F)

TAILSTRING(string, length) – Returns the tail portion of the string. (F)

TED(text_input) – Displays a table for text editing. (E)

TEXTSETUP(setupname) – Defines a setup for formatting a text document. (E)

TOKEN(string) – Parses an input string and returns the first token and the string with the token removed. (F)

TO_UNICODE(rdstring, externalcodepage) – Converts a raw data string encoded in an external code page to Unicode.(F)

$TYPECAST(type, syntax, size, decimals, value) – Converts a variable according to the arguments supplied. (F)

$UNPIC(picVal, mask) – Determines the original value submitted given a masked value produced by $PIC and the display mask that produced it. (F)

UNQUOTE(string) – Returns a string with the single quotation marks removed. (F)

UPPER_EBCDIC(string) – Converts a string to uppercase EBCDIC characters. (F)

UPPER_UNICODE(string) – Converts a string to uppercase Unicode characters. (F)

UPPERCASE(string) – Converts all lowercase characters in a string to uppercase characters. (F)

VAL_TO_LIT(string) – Converts a field to a string containing a token describing its value. (F)

VALID_NAME(name) – Determines if a given string satisfies the TIBCO Object Service Broker definition of an identifier. (F)

### Table Definitions and Data

BROWSER(tablespec) – Displays the contents of a TIBCO Object Service Broker table for viewing. (CE)

CLEARTABLE_APPL(table, select) – Deletes occurrences from a table or table instance. (E)

$CLRTAB(tablename, parm1, parm2, parm3, parm4) – Deletes (clears) the data rows from a table or table instance without reading the data rows. (C)

COPY_DATA(srctabspec, select, desttabspec, srclocation, destlocation, overwrite) – Copies data from one table or table instance to another table or table instance. (C)

COPYTABLE_APPL – Copies selected occurrences from a source table to a destination table. (E)

COUNTOCCURRENCES(table, selection) – Returns the number of occurrences that meet a selection criteria. (F)

DEFINE_TABLE(tbl_name) – Defines a new TIBCO Object Service Broker table or modifies an existing one. (E)

DELETE_DATA(tablespec, select, location) – Deletes the data from a table or table instance. (C)

DIFF_DATA(table1, field1, location1, selection1, table2, field2, location2, selection2, printresult) – Compares the data of two tables or table instances and lists the differences. (F)

EVENTTABLE – Returns the name of the table that activated the current derived field rule, trigger rule, or validation rule. (F)

FLDMGR(fieldname) – Adds fields to the global field dictionary. (E)

FORALLA(table, parm, selection, ordering) – Returns the first table occurrence that satisfies the selection criteria. Use if the value of every table parameter and every selection criterion is 99 or fewer characters long. (C)

@FORALLA(table, parm, selection, ordering) – Returns the first table occurrence that satisfies the selection criteria. Use if the value of any table parameter or of any selection criterion is 100 or more characters long. (C)

FORALLB(table) – Returns the next table occurrence that satisfies the selection criteria following a call to FORALLA. (C)

FORALLE(table) – Releases internally the resources used by FORALLA on a table. (C)

MOVTAB(tablename, segmentID) – Changes the segment number of a table. (CE)

INDEXCHK – Estimates the maximum number of data rows a table can contain before reaching the maximum index levels. (E)

NLS – Enables the database administrator to set code page values in translation tables. (E)

OBJECTMGR(tablespec) – Displays the contents of a table and enables a predefined set of commands that are unique to the table to operate on the display. (C)

PARMVALUE(parmname) – Returns the value of the parameter from the table that was accessed when the trigger or validation rule was activated. (F)

PARSE_TAM(string) – Breaks up an input string into a table specification, and optionally, the WHERE clause and the ORDERED clause of the corresponding table access statement. (C)

PROCESS_TABLE(tablespec, selection, ordering, processrule) – Provides specific processing for every occurrence in a table that is selected, ordered, or both selected and ordered. (C)

RETURN_MESSAGE – Returns the system error message whenever an exception is raised. RETURN_MESSAGE is a low-level tool that must be called immediately after an exception is trapped. (F)

SETNLSBIT(table, flag) – Sets the NLS bit for the specified table, in the RESERVED field of the TABLES table. (C)

SIMPLESELECT(selection) – Processes a selection string into a format that can be used by the FORALLA tool. (C)

SOE(tablespec) – Edits a single occurrence in a table. (CE)

STE(tablename) – Invokes the Table Editor. (CE)

STEBROWSE(input) – Views the contents of a TIBCO Object Service Broker table. (E)

TABLEPRINT(tablespec) – Prints the contents of a table or of a set of joined tables. (E)

### Trigger or Validation Rules

EVENTSCREEN – Returns the name of the screen that activates the current screen validation rule. (C)

EVENTTABLE – Returns the name of the table that activated the current derived field rule, trigger rule, or validation rule. (F)

PARMVALUE(parmname) – Returns the value of the parameter from the table that was accessed when the trigger or validation rule was activated. (F)