TIBCO iProcess™

Expressions and Functions Reference Guide

Software Release 11.1
February 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, The Power of Now, TIBCO ActiveMatrix, TIBCO Business Studio, TIBCO iProcess, TIBCO BusinessWorks, TIBCO PageBus are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, JMS and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. PLEASE SEE THE README.TXT FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2004-2012 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

About This Guide

This guide provides a complete reference to all data types, expressions and functions that you can use in iProcess procedures. It also contains information about TIBCO Business Studio-specific JavaScript classes.

-  indicates functions that are valid for the TIBCO iProcess Workspace (Windows) runtime environment.
-  indicates functions that are valid for the TIBCO iProcess Workspace (Browser) runtime environment.
-  indicates functions that are only used within the TIBCO Business Studio design time environment.

You should use this guide in conjunction with the following guides:

TIBCO iProcess Modeler

- *TIBCO iProcess™ Modeler - Basic Design*
- *TIBCO iProcess™ Modeler - Advanced Design*
- *TIBCO iProcess™ Modeler - Integration Techniques*

TIBCO Business Studio

- *TIBCO Business Studio™ Process Modeling User's Guide*
- *TIBCO Business Studio™ iProcess Developer's Guide*

How to Use This Guide

This guide is organized into the following logical sections:

- Chapter 1 gives detailed information on iProcess expressions that are used in various parts of a procedure definition.
- Chapter 2 describes how to use scripts in TIBCO Business Studio, and the differences between scripts in iProcess and TIBCO Business Studio.
- Chapter 3 provides summary information about iProcess functions.
- The remaining chapters, organized by type, provide detailed reference information on each available function.

Target Audience

This guide is aimed at **procedure definers**, who convert business processes into TIBCO iProcess procedures or TIBCO Business Studio processes, and **systems integrators**, who integrate processes with other applications.

The guide assumes that you are familiar with TIBCO iProcess procedures or TIBCO Business Studio processes and how to define them. If you are not, refer to the appropriate guides listed previously.

Changes from the Previous Issue

This issue contains minor updates.

Where You Can Find More Information

You can find more information about the *TIBCO iProcess™ Workspace* and the *TIBCO iProcess Modeler* from the following sources, which are supplied with the TIBCO iProcess Workspace (Windows) software (in the **\docs** directory):

- *Installation Guide*, which explains how to install, upgrade and remove the software.
- *Release Notes*, which provide information about new and changed features, known issues and closed issues for this release.
- *Readme* file, which provides any last-minute and version-specific information that could not be included in the main documentation.
- There is also a useful resource, **http://power.tibco.com**, that delivers technical content to the TIBCO user community. This site has been developed to foster an open forum where users of TIBCO products can find valuable information, example projects and resources for those projects, and exchange ideas with other users. Entry to this site requires a username and password. If you do not have a username, you can request one.

Online help is also available within the iProcess™ Workspace (Windows) and iProcess™ Modeler.

Documentation Conventions

Because this guide covers both Windows, UNIX and Linux versions of the TIBCO iProcess™ Engine, this guide uses the Windows convention of a backslash (\). The equivalent pathname on a UNIX or Linux system is the same, but using the forward slash (/) as a separator character.

UNIX or Linux pathnames are occasionally shown explicitly, using forward slashes as separators, where a UNIX/Linux-specific example or syntax is required.

Any references to UNIX in this guide also apply to Linux unless explicitly stated otherwise.

The following conventions are used throughout this guide.

Convention	Description
SWDIR	Indicates the iProcess system directory where the TIBCO iProcess Engine is installed. **Example:** If *SWDIR* is set to **swserver****staffw_nod1** then the full path to the **swutil** command would be: • on a Windows server (on the **c:** drive): **c:\swserver\staffw_nod1\bin\swutil** • on a UNIX server: /**swserver/staffw_nod1/bin/swutil**, or $SWDIR**/bin/swutil** **Note: On a UNIX system, the environment variable $SWDIR should be set up to point to the iProcess system directory for the root and iProcess Administrator users.**
italics	Indicates emphasis, variables and manual titles.
`monospace text`	Indicates code samples, commands and their options, directories and filenames. Any text that you must enter from the keyboard is displayed as monospace text.
`monospace italic text`	Indicates variables in commands.

Convention	Description
{ }	Indicates a set of choices in a syntax line. The braces should not be entered.
[ ]	Indicates optional items in a syntax line. The brackets should not be entered. **Example:** SHOW_ALL_ATTRIBUTES [attribute]
|	Indicates mutually exclusive choices in a syntax line i.e. you enter only one of the given choices. You should not enter the symbol itself.

Chapter 1 TIBCO iProcess Expressions

This chapter gives detailed information on iProcess Expressions that are used in various parts of a procedure definition. iProcess uses them when running cases of the procedure to evaluate results or resolve conditions. An expression consists of:

- **constant** values (for example, **2.3** or **"Yes"**) and/or
- **field** names (for example, **STARTDATE** and **MEMOFIELD**) combined by
- **operators** (for example, the addition operator, **+**), and
- **functions** (for example, **STRLEN(NAME)**). A summary of functions can be found in the Functions Summary on page 41.

Topics

Data Types

A field or constant has a specific **type** that determines its allowable values. Operators and functions work on items of certain types and yield a result of a specific type.

Expressions can include items of any of the field data types available in iProcess:

Field Type	Description
Text	Strings of text characters in the Staffware Internal Character Set (SICS), length 0 to 255 characters. A text constant consists of the characters enclosed in quotes, e.g "Peter". (To include the quote character itself in a text constant, include it twice, `""`.)
Memo	Standard text memos. For example, you can create the following expressions: • memofield := text • memofield := memofield • memofield := memofield + text • textfield := memofield • textfield := memofield + text For the last two examples, you need to be aware that truncation of the memo data can occur because the maximum length of a text field is 255 characters. If you are adding text to a memo, you can use the SPECIALCHARS /n option to add the text on a new line, for example: expr: memofield := memofield + memofield memofield := memofield + specialchars ("\nAdd text on a new line\n") You can only use the following functions with memofields: • STRLEN • SEARCH and RSEARCH

Field Type	Description
Numeric	Numbers in the range –99,999,999,999,999 to 999,999,999,999,999, including decimals. Numeric constants are written using the usual digits, with a period for decimal numbers, and a - sign for negative numbers, for example -2.6. (Your system may be set up to use a comma as a decimal separator, for example -2,6.)
Date	Dates in the range 1/1/0 to 31/12/2999. A date constant is written !DD/MM/YYYY!, for example !06/09/1997! is 6th September, 1997. (Your system may be set up for a different format, for example !MM/DD/YYYY!.)
Time	Times, resolution to minutes. A time constant is written #HH:MM# (24-hour clock format - range #00:00# to #23:59#.)

In addition to the field data types, some further types can appear in iProcess expressions; these are:

Type	Description
Boolean	Result of a relational/logical operation (i.e. true or false). Note that a Boolean constant cannot be entered.
Date Offset	A constant used in expressions to modify date types, written @day/week/month/year@, so @2/0/1/0@ will increment a date by 1 month and 2 days. (Note that the numbers must be constants; to increment a date by the contents of fields, there is a function CALCDATE available.)
Vartype	Used to handle variable data types for the SELECTVAL and SWITCHVAL functions. A vartype will return or accept as input any data type currently available in iProcess.

An expression as a whole has a particular type; the allowed expression types are: Text, Numeric, Date, Time and Boolean.

Use of Expressions

iProcess expressions can appear in a number of different places in a procedure definition. A condition expression must return type Boolean (i.e. it is true or false); others return different types according to the context. The following table lists all the locations with the data types of the expressions:

Location	Data type
Validations section in Required or Optional fields	field type
Calculations section in Calculated or Hidden fields	field type
Conditions section in Calculated or Hidden fields	Boolean
Conditions in form text	Boolean
Conditions in scripts	Boolean
Deadline date expression	date
Deadline time expression	time
Deadline condition	Boolean
Action condition	Boolean
Restricted procedure access attributes	Boolean
Form or Field Command	any (return value thrown away)
Script statement	any (return value thrown away)
Sub-Procedure Input Parameters	field type
Sub-Procedure Output Parameters	field type

Defining Expressions

This section describes the different components of an expression and how they are combined (except for functions, which are described later). These components are:

- Field Names
- Constants
- User Attributes
- iProcess Variables
- System Values
- Operators

Field Names

There are two types of fields:

- single instance fields
- array fields

These are entered as marked in forms (upper or lower case). Note that Table field names must include the tag name and the Table field name, for example CUST:NAME.

See “Creating Fields and Forms” in the *TIBCO iProcess Modeler - Basic Design* guide for more information on single instance fields and “Using Arrays” in the *TIBCO iProcess Modeler - Advanced Design* guide for more information on array fields.

Constants

These are entered as described under the relevant type in Data Types on page 8.

User Attributes

These yield the attribute values for various users, as set up by the System Administrator. They have the same types as the attributes.

Value	Description
SW_GROUP:*attribute*	The group queue from which the work item was chosen **Note: This field is not available to the TIBCO iProcess™ Script Server Plug-in.**
SW_STARTER:*attribute*	Starter of the case
SW_USER:*attribute*	Current user **Note: This field is not available to the TIBCO iProcess Script Server Plug-in.**

Examples

SW_USER:DESCRIPTION yields the description (or long name) of the current user, and is of type text.

The Boolean expression SW_USER:GRP_MYGROUP = "Yes" returns TRUE if the current user belongs to the group MYGROUP. This is useful if you want to restrict case start access to users who belong to a particular group. To do this, type the expression in the **Expressions** box in the **Procedure Access - Start Case** dialog. See “Restricting Case Starts” in the *TIBCO iProcess Modeler - Procedure Management* guide for more information.

iProcess Variables

The following iProcess variables are defined.

Variable	Description
$RETURN	(any type) $RETURN can only be used in scripts. A value may be assigned as a result of a script being executed or $RETURN may be used in an expression. When a script has finished executing, the value of this variable is used as the return value of the script. If the value of this variable has not been assigned during the execution of the script, the return value of the script is the value of the last expression executed within the script. See SCRIPT on page 155 for more information about the SCRIPT function.
$ARG*n*	(text) $ARG*n* can only be used in scripts. Use $ARG*n* to pass parameter values into a script. See SCRIPT on page 155 for more information about the SCRIPT function. **Note: The $ARGn variable is treated as a string by iProcess. This means if you pass a numeric value to $ARG*n*, you must convert the string back to a numeric value within the script. For example, accnum:=num($ARG1).**
$IP*n*	(any type) $IP*n* can only be used to change the value of input parameters in a sub-case using an event step in the parent process. $IP*n* refers to a sub-procedure input parameter where *n* is a positive integer that is automatically assigned to the sub-procedure input parameter by iProcess. $IP*n* inherits the type of the sub-procedure input parameter. For more information, see *TIBCO iProcess Modeler - Integration Techniques*.
$IPT*n*	(any type) $IPT*n* can only be used to change the value of input parameters in a sub-case started from a dynamic sub-proceedure call. This is done using an event step in the parent process. $IPT*n* refers to a sub-procedure input parameter derived from a sub-procedure parameter template where *n* is a positive integer that is automatically assigned to the sub-procedure input parameter by iProcess. $IPT*n* inherits the type of the sub-procedure template input parameter.
$OP*n*	(any type) $OP*n* can only be used in output parameter scripts. $OP*n* refers to a sub-procedure output parameter where *n* is a positive integer that is automatically assigned to the sub-procedure output parameter by iProcess. $OP*n* inherits the type of the sub-procedure output parameter. For more information, see *TIBCO iProcess Modeler - Integration Techniques*.

Variable	Description
$OPT*n*	(any type) $OPT*n* can only be used in sub-procedure output parameter scripts. $OPT*n* refers to a sub-procedure output parameter derived from a sub-procedure parameter template where *n* is a positive integer that is automatically assigned to the sub-procedure output parameter by iProcess. $OPT*n* inherits the type of the sub-procedure template output parameter.

System Values

The following system values are defined.

Value	Description
SW_ANYTHING	(any type) any value. This may only be used as validation for a field marking. It means that as well as other values specified in the validations, you may enter any value into the field.
SW_BLANK	(text) a null text constant, i.e. `""`. (Note that this is not the same as SW_NA - see below.)
SW_CASEDESC	(text) case description.
SW_CASENUM	(numeric) case number.
SW_CASEREF	(text) case reference (*pp-nn*).
SW_DATE	(date) current system date.
SW_GEN_IDX	(numeric) generic array field if an array field's individual index is unassigned.
SW_HOSTNAME	(text) host name for the procedure.
SW_IP_VALUE	(numeric) work item priority value. **Note: Refer to "Using Work Queue Parameter Fields" in the *TIBCO iProcess Modeler - Advanced Design* guide for more information about the use of the SW_IP_* work item priority fields.** **Note: This field is not available to the TIBCO iProcess Script Server Plug-in.**
SW_IP_INCREMENT	(numeric) value to be added to the work item's priority value SW_IP_VALUE whenever the increment period SW_IP_INCPERIOD expires.
SW_IP_NUMINC	(numeric) number of SW_IP_VALUE increments to be added to the work item's priority value SW_IP_VALUE.
SW_IP_INCPERIOD	(numeric) time period, in units specified in SW_IP_PERIODTYP, which must expire before the work item's priority value SW_IP_VALUE is incremented.
SW_IP_PERIODTYP	(text) unit of measure of the increment period SW_IP_INCPERIOD.

Value	Description
SW_NA	(any type) Not Assigned - a field has no value.
SW_NODENAME	(text) node name of the system.
SW_PRODESC	(text) procedure description.
SW_PRONAME	(text) name of the procedure.
SW_QPARAM*n*	(text) application specific data which can be used in Work Queue Manager to display, sort or filter work queues. **Note: Four fields are available: SW_QPARAM1 to SW_QPARAM4. Refer to "Organizing Your Work Item Lists" in the *TIBCO iProcess Workspace (Windows) User's Guide* for more information about the use of these fields.**
SW_QRETRYCOUNT	(numeric) number of times that a message in a message queue has failed. The field's value is **0** the first time a message is processed, and is incremented each time the message fails. For example, if a **BG** process is processing a message and **SW_QRETRYCOUNT = 2**, this means that the **BG** is attempting to process the message for the third time. **Note - The SW_QRETRYCOUNT field only returns a meaningful value when it is used during the processing of a message by a BG process. If it is used in any other circumstance (for example, displayed on a form) it will return SW_NA. If you want to display the value in a form or use it elsewhere in the procedure you must first use an EAI Script step to assign it to another field, as part of the same transaction.**
SW_STEPDESC	(text) description of the step.
SW_STEPNAME	(text) name of the step.
SW_TIME	(time) current system time.

The following system values are relevant to sub-procedures:

Value	Description
SW_MAINCASE	Top level procedure's case number.
SW_MAINPROC	Top level procedure's name.
SW_MAINHOST	Host where top level procedure resides.

Value	Description
SW_PARENTCASE	Parent procedure's case number.
SW_PARENTPROC	Parent procedure's name.
SW_PARENTHOST	Host where parent procedure resides.
SW_PARENTREF	Internal information on parent.

The following system values are relevant to case prediction.

Value	Description
SW_ARRIVALDATE	(date) The date when case prediction has calculated the step will arrive in the queue.
SW_ARRIVALTIME	(time) The time when case prediction has calculated the step will arrive in the queue.
SW_LEAVEDATE	(date) The date when case prediction has calculated the step will leave the queue.
SW_LEAVETIME	(time) The time when case prediction has calculated the step will leave the queue.

Operators

iProcess supports a set of operators that can be used in expressions. The table below shows how different data types can be combined with each of the operators.

The table gives the **operator**, the **allowable types** (or type pairs), the **result** type and the **precedence** of the operator.

The type pair any/any is used to represent a pair of the SAME type from numeric, text, date, time. The iProcess Attachment type is treated as type text for the purposes of this table.

In mixed date/numeric expressions, numerics represent days; in mixed time/numeric expressions, numerics represent minutes.

The highest **precedence** is 0, the lowest is 6. Operations of higher precedence in mixed expressions are calculated first, so 2 + 3 * 4 results in 14, not 20, as multiplication has a higher precedence than addition.

You can force precedence with **parentheses**, so (2 + 3) * 4 results in 20, as the part of the calculation inside the brackets is calculated first.

Operators of the same precedence are **left-associated** (except for assignment), i.e. 1 – 2 + 3 is read as (1 – 2) + 3 and results in 2, not –4.

Operator	Allowable types	Result	Precedence
– (negation)	numeric	numeric	0
^ or ** (exponentiation)	numeric/numeric	numeric	1
* (multiplication)	numeric/numeric	numeric	2
/ (division)	numeric/numeric	numeric	2
+ (addition)	numeric/numeric	numeric	3
	date/date-offset	date	3
	date/numeric	date	3
	text/text	text	3
	time/numeric	time	3
	any/SW_NA	any	3

Operator	Allowable types	Result	Precedence
– (subtraction)	numeric/numeric	numeric	3
	date/date	numeric	3
	date/date-offset	date	3
	date/numeric	date	3
	time/numeric	time	3
	time/time	numeric	3
= (equality)	any/same	Boolean	4
	any/SW_NA	Boolean	4
	text/SW_BLANK	Boolean	4
<> (inequality)	any/same	Boolean	4
	any/SW_NA	Boolean	4
	text/SW_BLANK	Boolean	4
> (greater than)	any/same	Boolean	4
< (less than)	any/same	Boolean	4
>= (greater or equal)	any/same	Boolean	4
<= (less or equal)	any/same	Boolean	4
AND (logical)	Boolean/Boolean	Boolean	5
NOT (logical)	Boolean/Boolean	Boolean	5
OR (logical)	Boolean/Boolean	Boolean	5
:= (assignment)	any/same	any	6
	numeric/Boolean	numeric	6

Allowable operations may be determined from the table above. Allowable types are named, and if any type is allowed, it is listed as **any**. Some operators allow any type, but require that that the types being compared are the same (this is listed as **any/same**). The result of most operations is obvious (particularly for numerics); the following section defines the results of certain specific actions.

Although SW_BLANK can be assigned to any type from an iProcess field's validation list, its type in the expression evaluator is text, therefore you can only use SW_BLANK with text types.

Division (/)

Division of a number by zero results in SW_NA.

Addition (+)

Addition of strings results in concatenation in left to right order.

Addition of a date and date offset results in a date, adjusted accordingly.

Addition of a date and a numeric (number of days) results in a date (determined according to the current working days configuration).

Addition of a time and a numeric (number of minutes) results in a time.

Addition of any value to SW_NA gives SW_NA, except for a text value which gives that value.

Subtraction (–)

Subtraction of a date offset from a date results in a date, adjusted accordingly.

Subtraction of a date from a date results in the number of days (either positive or negative). (Note: this value is determined according to the current working days setting.)

Subtraction of a numeric (number of days) from a date results in a date (determined according to the current working days configuration).

Subtraction of a numeric (number of minutes) from a time results in a time.

Subtraction of a time from a time results in the number of minutes difference (positive or negative).

Equality (=)

Equality of an iProcess identifier of any type with SW_NA results in true if the value has not been defined, false if it has been defined.

Text comparisons are case *in*sensitive.

Inequality (<>)

Follows the same rules as Equality, but returns the opposite truth value.

Relational operators (<, >, <=, >=)

Dates and times are compared chronologically.

Text items are compared using the Staffware Internal Character Set (SICS) collating sequence, except that comparisons are case insensitive.

Assignment (:=)

An iProcess field may have its value assigned from the result of an expression. Note that both sides of the assignment operator must be of the same type.

As a special case, the result of a Boolean expression may be assigned to a numeric variable; TRUE yields 1 and FALSE yields 0.

An assignment expression returns a value equal to the assigned value; this enables multiple assignments to be made and assignments to be used in function calls, for example:

```
NUM1 := NUM2 := 0
```

sets the values of fields NUM1 and NUM2 to zero.

```
LEN := STRLEN (TEXT1 := TEXT2)
```

copies the string in field TEXT2 into TEXT1, then puts the string's length into numeric field LEN.

Examples

This section illustrates some expressions as they might be used in iProcess procedures without functions. (See the next section for functions.)

- Test that the field of an iProcess Table record variable has defined contents:

```
TABTAG:TABFIELD <> SW_NA
```

- Test if the user of this procedure is "JOHN":

```
SW_USER:NAME = "JOHN"
```

- Calculate the total price based on number of items, unit cost and VAT rate:

```
(NUM_UNITS * UNIT_PRICE) * VAT_RATE
```

- Test if an order exceeds a customer's credit limit, defined in an iProcess Table:

```
TOTAL_VALUE >= CUST_REC:CREDIT_LIM
```

- Produce a composite name from components, for example "Mr. John Smith":

```
EMPLOYEE:SALUTATION + " " + EMPLOYEE:FNAME
+ " " + EMPLOYEE:LNAME
```

- Calculate a date 3 months from now:

```
SW_DATE + @0/0/3/0@
```

- Test if someone is old enough in an application procedure:

```
(SW_DATE - DATE_OF_BIRTH) > ((AGE_LIMIT * 365) +
(AGE_LIMIT / 4))
```

Using Regular Expressions

Regular expressions may be included in filter criteria expressions. They must be in the following format:

```
constant ? "regular expression"
```

where:

- *constant* is a constant value or field name
- ? is a special character signifying that a regular expression follows (interpreted as an equality operator)
- *regular expression* is any valid regular expression (enclosed in double quotes)

Examples

```
"abcdefg"?"abc*" (result = true)
"abcdefg"?"a*d*g" (result = true)
field1?"abc*[0-9]" (result = true, assuming field1 has the value
"abcd5")
field1?"[a-z]bcd[0-9]" (result = true, assuming field1 has the
value "abcd5")
```

A regular expression (RE) specifies a set of character strings. A member of this set of strings is "matched" by the RE. The REs allowed are:

The following one-character REs match a single character.

- An ordinary character (not one of those discussed in item 2 below) is a one-character RE that matches itself.
- A backslash (\) followed by any special character is a one-character
- RE that matches the special character itself. The special characters are:
  - ., *, [, and \ Period, asterisk, left square bracket, and backslash, respectively. These are always special, except when they appear within square brackets ([ ]; see Item 4 below).
  - ^ Caret or circumflex, which is special at the beginning of an entire RE, or when it immediately follows the left bracket of a pair of square brackets ([ ]) (see Item 4 below).
  - $ Dollar sign, which is special at the end of an entire RE. The character used to bound (i.e., delimit) an entire RE, which is special for that RE.
- A period (.) is a one-character RE that matches any character except new-line.
- A non-empty string of characters enclosed in square brackets ([]) is a one-character RE that matches any one character in that

- string, with these additional rules:
  - If the first character of the string is a circumflex (^), the one character RE matches any character except new-line and the remaining characters in the string. The ^ has this special meaning only if it occurs first in the string.
  - The minus (-) may be used to indicate a range of consecutive characters. For example, [0-9] is equivalent to [0123456789].The minus sign loses this special meaning if it occurs first (after an initial ^, if any) or last in the string.
  - The right square bracket (]) does not terminate such a string when it is the first character within it (after an initial ^, if any). For example, [ ]a-f] matches either a right square bracket (]) or one of the ASCII letters a through f, inclusive.
  - The special characters ., *, [, and \ stand for themselves within such a string of characters.

The following rules may be used to construct REs from one-character REs:

- A one-character RE is a RE that matches whatever the one-character RE matches.
- A one-character RE followed by an asterisk (*) is an RE that matches zero or more occurrences of the one-character RE. If there is any choice, the longest, left most string that permits a match is chosen.
- A one-character RE followed by \{*m*\}, \{*m*,\}, or \{*m*,*n*\} is an RE that matches a range of occurrences of the one-character RE. The values of *m* and *n* must be non-negative integers less than 256:
  - \{*m*\} matches exactly m occurrences;
  - \{*m*,\} matches at least m occurrences;
  - \{*m*, *n*\} matches any number of occurrences between *m* and *n* inclusive.

Whenever a choice exists, the RE matches as many occurrences as possible:

- The concatenation of REs is an RE that matches the concatenation of the strings matched by each component of the RE.
- An RE enclosed between the character sequences \(and \) is an RE that matches whatever the unadorned RE matches.
- The expression *n* matches the same string of characters as was matched by an expression enclosed between \(and \) earlier in the same RE. Here n is a digit; the sub-expression specified is that beginning with the nth occurrence of \(counting from the left. For example, the expression ^\(.*\)\1$ matches a line consisting of two repeated appearances of the same string.

An RE may be constrained to match words:

- \< constrains an RE to match the beginning of a string or to follow a character that is not a digit, underscore, or letter. The first character matching the RE must be a digit, underscore, or letter.
- \> constrains an RE to match the end of a string or to precede a character that is not a digit, underscore, or letter.

An entire RE may be constrained to match only an initial segment or final segment of a line (or both):

- A circumflex (^) at the beginning of an entire RE constrains that RE to match an initial segment of a line.
- A dollar sign ($) at the end of an entire RE constrains that RE to match a final segment of a line.
- The construction ^entire RE$ constrains the entire RE to match the entire line.

The null RE is equivalent to the last RE encountered.

Concerning the use of '?' vs. '=', you should only use the '?'character when matching regular expression patterns. If comparing an integer value or string, it is more efficient to use the equality operator, '='. For Example:

```
oWorkQ.WorkItems.FilterExpression =
"SW_PRONAME=""LOAN"""
```

Chapter 2 TIBCO Business Studio Scripts

This chapter gives detailed information on the JavaScript classes that are used in various parts of TIBCO Business Studio.

Topics

Use of Scripts

There are several places you can enter JavaScript in TIBCO Business Studio.

The JavaScript script grammar is only available with selected destination environments, and when the Solution Design capability is selected:

Location	How to Access...	Notes
Script Task	In the Properties view, on the **General** tab.	
Conditional Sequence Flow	In the Properties view, on the **General** tab.	Limited to one line that evaluates to Boolean.
Auditing Scripts (Initiated, Completed, Timeout, Cancel)	In the Properties view, on the **Scripts** tab.	Initiated and Completed only are supported in iProcess, and limited to only one line that must evaluate to a string.
User Task Scripts (Open, Close, Submit)	In the Properties view, on the **Scripts** tab.	
Loop Scripts *(not applicable to iProcess)*	In the **Loops** tab, when **Standard Loop** or **Multiple Instance Loop** is selected on the **General** tab.	Limited to one line that evaluates to Boolean.

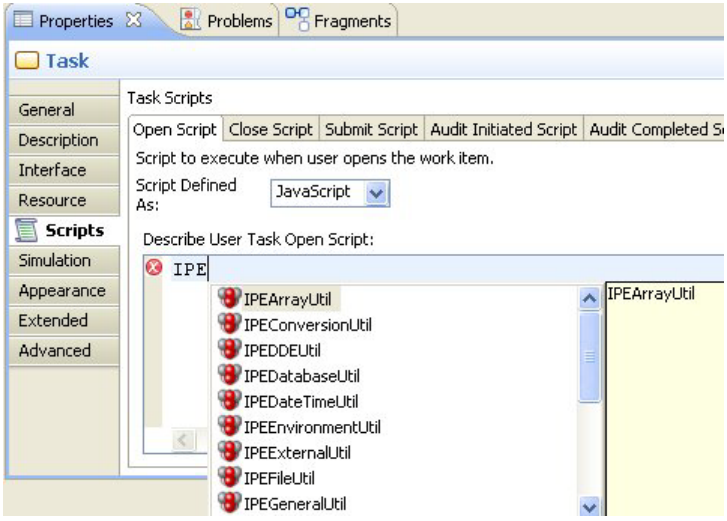
Location	How to Access...	Notes
Catch Timer Events	In the Properties view, on the **General** tab.	Limited to two statements. If you specify only one statement, it must evaluate to either a Date or a Time. If you specify two statements, one must evaluate to a Date and the other to a Time. The script area cannot be empty if you want to export or deploy to the iProcess Engine (it can however be empty for a destination environment that includes the iProcess Modeler destination component).

Content Assist

Content assist is provided in the script entry areas of TIBCO Business Studio. This allows you to quickly enter the following:

- iProcess script functions
- iProcess system fields
- templates for common JavaScript constructs
- process data

For example, if you enter **IPE**, then press **Ctrl** + **Space**, you can view the iProcess functions available:



In addition to content assist, TIBCO Business Studio provides full script validation using the Eclipse Problems view.

Differences Between TIBCO Business Studio and TIBCO iProcess

This section lists areas in which the iProcess use of expressions and functions differs from TIBCO Business Studio script usage.

Data Types

TIBCO Business Studio supports the data fields available in iProcess as follows:

TIBCO Business Studio Data Type	iProcess Data Type	Notes
String	Text	
String with no maximum length	Memo	
Decimal Number (Length 10, Decimal Places, 2)	Numeric (Length 11, Decimal Places, 2)	
Integer Number	Numeric with zero decimal places	
Boolean	Decimal with length of one and zero decimal places	
Date	Date	
Time	Time	
Date Time		In iProcess, TIBCO Business Studio datetime data types are broken down in date and time fields with **_D** and **_T** appended (for example, **myDt_D** and **myDt_T**).
Performer	Text (Length 255)	

TIBCO Business Studio Data Type	iProcess Data Type	Notes
n/a	VarType	TIBCO Business Studio Strings, Decimal Numbers, Integer Numbers, Date, Time, and the Date or Time portion of a Date Time can all be passed to a VarType parameter.
n/a	commaSeparatedNumeric	
n/a	composite	

User Attributes

To obtain attributes of users like Starter, Group, and User, you can use the following Util classes. For more information, see TIBCO Business Studio JavaScript Classes on page 183.

iProcess Value	TIBCO Business Studio Equivalent
SW_GROUP:*attribute*	IPEGroupUtil.GETATTRIBUTE(String);
SW_STARTER:*attribute*	IPEStarterUtil.GETATTRIBUTE(String);
SW_USER:*attribute*	IPEUserUtil.GETATTRIBUTE(String);

Regular Expressions

Regular expressions are not supported in TIBCO Business Studio.

iProcess Variables

The variables $RETURN and $ARG*n* are not supported. The variables $OP*n* and $OPT*n* are supported as follows:

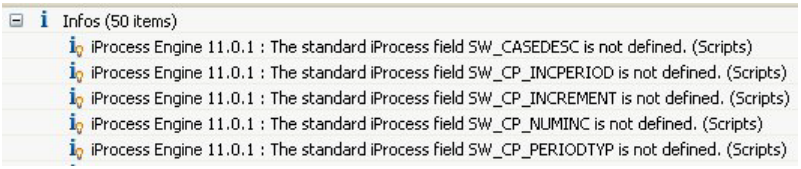
iProcess Value	TIBCO Business Studio Equivalent
$OP*n*	This variable is not used, however sub-process parameter names that you use are automatically converted to the $OP*n* syntax. For an explanation of this variable, see iProcess Variables on page 13.

iProcess Value	TIBCO Business Studio Equivalent
$OPT*n*	This variable is not used, however sub-process template parameter names that you use are automatically converted to the $OPT*n* syntax. For an explanation of this variable, see iProcess Variables on page 13.

System Values

System values are expressed in two different ways in TIBCO Business Studio, depending on whether they are read only or read/write:

- Read only fields (ones that should not be modified by the user) are listed in the **IPESystemValues** class. For example, SW_DATE should not be modified by the user; it is used to display the system date. This can be expressed in TIBCO Business Studio using **IPESystemValues.SW_DATE**.
- Fields that are read/write display an informational message in the problems view:



  By right-clicking one of these informational messages and selecting **Quick Fix**, you can create the system value as data field that you can modify if necessary.

The following table shows the system fields.

System Field Name	Read only or read/write	Data type	Length
SW_CASEDESC	read/write	String	24
SW_CASENUM	read only	Integer Number	15
SW_CASEREF	read only	String	20
SW_CP_INCPERIOD	read/write	Integer Number	4
SW_CP_INCREMENT	read/write	Integer Number	4
SW_CP_NUMINC	read/write	Integer Number	3

System Field Name	Read only or read/write	Data type	Length
SW_CP_PERIODTYP	read/write	String	1
SW_CP_VALUE	read/write	Integer Number	3
SW_DATE	read only	Date	n/a
SW_GEN_IDX	read/write	Integer Number	6
SW_HOSTNAME	read only	String	24
SW_IP_INCPERIOD	read/write	Integer Number	4
SW_IP_INCREMENT	read/write	Integer Number	4
SW_IP_NUMINC	read/write	Integer Number	3
SW_IP_PERIODTYP	read/write	Integer Number	1
SW_IP_VALUE	read/write	Integer Number	3
SW_PRODESC	read only	String	24
SW_PRONAME	read only	String	8
SW_QRETRYCOUNT	read only	Integer Number	15
SW_STEPDESC	read only	String	24
SW_STEPNAME	read only	String	8
SW_TIME	read only	Time	n/a
SW_MAINCASE	read only	Integer Number	10
SW_MAINHOST	read only	String	24
SW_MAINPROC	read only	String	8
SW_PARENTCASE	read only	Integer Number	10
SW_PARENTHOST	read only	String	24
SW_PARENTPROC	read only	String	8
SW_PARENTREF	read only	String	64

System Field Name	Read only or read/write	Data type	Length
SW_NODENAME	read only	String	24

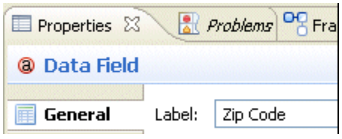
Operators

TIBCO Business Studio uses the standard JavaScript operators. The following table shows those operators that differ from the operators in iProcess.

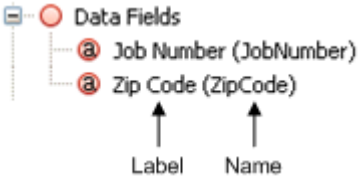
TIBCO Business Studio	iProcess
==	= (equality)
!=	!= or <> (inequality)
& or &&	AND (logical)
!	NOT (logical)
| or ||	OR (logical)
=	:= (assignment)
^ (** not supported)	^ or ** (exponentiation)

Labels and Names

When business analysts (using the Business Analysis capability) create process objects such as data fields, task names, and so on, they assign the objects labels that may contain spaces or non-alphanumeric characters.



With the Solution Design capability selected, the Label as well as the Name is displayed. For example:



In scripts, content assist only shows Names, and you must use the Name to reference process data. If you migrate a process from an earlier version, the migration XSLT creates labels from the name.

TIBCO Business Studio-Specific Classes

There is a group of JavaScript classes that are only used with TIBCO Business Studio. These are listed in TIBCO Business Studio JavaScript Classes on page 183.

For example, when using process names in TIBCO Business Studio scripts, use the method **iPEProcessNameUtil.GETPROCESSNAME** to convert valid TIBCO Business Studio process names into equivalent iProcess procedure names. Doing so means that long process names are truncated as they would be upon deployment to iProcess. This ensures that a sub-process call task or a dynamic sub-process works as expected upon deployment. Similarly, use the **iPETaskNameUtil** method when referring to task names.

Chapter 3 Functions Summary

Functions defined in this guide can be used in expressions anywhere that a constant or field appears, provided the return type of the function is correct.

A function call consists of the **name** of the function (upper or lower case) followed by the function **arguments** separated by commas and enclosed in brackets, for example:

```
DATE (DAY, LASTMONTH, 2001)
```

In TIBCO iProcess Modeler, the syntax for the previous example is:

```
IPEDateTimeUtil.DATE(Integer,Integer,Integer);
```

A function with no arguments has just the brackets, for example:

```
DDETERMALL ()
```

If your iProcess system is set up to use a comma as a decimal separator, you must separate arguments with a comma followed by a space to avoid ambiguity, for example str(2, 2) instead of str(2,2).

In general, the function performs an operation on the arguments and returns a value. In some cases, an argument must be the name of a field that will receive a return value. Otherwise, each argument to a function is itself an expression - a single constant or fieldname, items combined by operators, or function calls.

The following groups of functions are provided. In TIBCO iProcess Modeler, JavaScript classes are provided for the main function types:

Function Type	Description	See
Conversion Functions	Convert data to different formats.	page 43
Environment Functions	Get and set environment data.	page 53
File Functions	Manipulate files.	page 97
Date and Time Functions	Get and set date and time data.	page 105
String (Text) Functions	Manipulate text strings.	page 119

Function Type	Description	See
Functions to Call External Programs	Call external programs on the server or on a TIBCO iProcess Workspace (Windows).	page 125
Validation Functions	Add data from a file to a field's validation list.	page 135
Dynamic Data Exchange (DDE) Functions	Use Dynamic Data Exchange (DDE) to transfer data between two Windows applications while they are running.	page 141
Calling Scripts	Call iProcess scripts.	page 153
Database Functions	Write fields within a work item to a table in the iProcess database on the server	page 157
Procedure Functions	Control the processing of cases.	page 161
Array Functions	Reference array elements.	page 171
General Utility Functions	General utility functions.	page 177

Chapter 4 Conversion Functions

The following functions can be used to convert data to different formats depending on your requirements.

Function	Usage	Description	See
DATESTR		Convert a date to a string	page 44
NUM		Convert a string to a number	page 45
SPECIALCHARS		Include non-printing characters in a text string	page 46
STR		Convert a number to a string	page 47
STRCONVERT		Convert a text string	page 48
STRTOLOWER		Convert text to lower case	page 49
STRTOUPPER		Convert text to upper case	page 50
TIMESTR		Convert a time to a string	page 51

DATESTR

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Convert a date to a text string.

Syntax

```
datestr (date)
```

where *date* is the date to be converted.

Returns

A text string containing the text equivalent of the date passed, in the format DD/MM/YYYY (or otherwise according to your system configuration).

Example

TIBCO iProcess Modeler:

```
datestr(!20/01/2009!)
```

returns 20/01/2009

TIBCO Business Studio:

```
Field = IPEConversionUtil.DATESTR(IPESystemValues.SW_DATE);
```

This example uses the SW_DATE function to get today's date, converts the date to a string, and assigns the string to **Field** (a string data field).

NUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the numeric value of the text string passed. An invalid string will result in SW_NA.

Syntax

```
num (text)
```

where *text* is a text string.

Returns

Numeric value of *text*.

Examples

TIBCO iProcess Modeler:

```
num ("123")
```

returns 123

TIBCO Business Studio:

```
IPEConversionUtil.NUM("123");
```

SPECIALCHARS

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Include non-printing characters in a text string.

Syntax

```
specialchars (text)
```

where *text* is a text string which may include any number of the following sequences (plus ordinary text if required):

Sequence	Meaning
\n	newline
\r	carriage return
\t	tab
nnn	the character with decimal code *nnn* (must be 3 digits, so include leading zeros if required)
\\	a literal backslash \

Returns

The resulting *text* string.

Examples

TIBCO iProcess Modeler:

```
specialchars(“Your test results are\r\n English=80 \r\n Maths=90")
```

returns

```
Your test results are
English=80
Maths=90
```

TIBCO Business Studio:

```
IPEConversionUtil.SPECIALCHARS("Your test results are\r\n
English=80 \r\n Maths=90");
```

STR

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the textual equivalent of the number passed, to the specified number of decimals.

Syntax

```
str (number, decimals)
```

where:

- *number* is a numeric value.
- *decimals* is the required number of decimals, as a numeric.

Returns

The equivalent text string.

Examples

TIBCO iProcess Modeler:

```
str (2.3, 2)
```

returns "2.30"

```
str (2.3, 0)
```

returns "2"

TIBCO Business Studio:

```
IPEConversionUtil.STR(2.3,2);
IPEConversionUtil.STR(2.3,0);
```

STRCONVERT

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Convert a text string.

Syntax

```
strconvert (text, operation)
```

where:

- *text* is the string to be converted.
- *operation* (numeric) is the type of conversion. Values may be added for combinations of operations:

Value	Conversion Type
1	delete all spaces
2	delete all leading spaces
4	delete all trailing spaces
8	reduce sequences of multiple spaces to single spaces
16	convert to lowercase
32	convert to uppercase

Returns

The *text* string after conversion.

Examples

TIBCO iProcess Modeler:

```
strconvert ("test", 32)
```

returns "TEST"

TIBCO Business Studio:

```
IPEConversionUtil.STRCONVERT("test",32);
```

STRTOLOWER

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Convert text to lower case.

Syntax

```
strtolower (text)
```

where *text* is the text string to be converted.

Returns

The *text* string after conversion.

Examples

TIBCO iProcess Modeler:

```
strtolower ("TEST")
```

returns "test"

TIBCO Business Studio:

```
IPEConversionUtil.STRTOLOWER("TEST");
```

STRTOUPPER

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Convert text to upper case.

Syntax

```
strtoupper (text)
```

where *text* is the text string to be converted.

Returns

The *text* string after conversion.

Examples

TIBCO iProcess Modeler:

```
strtoupper ("test")
```

returns "TEST"

TIBCO Business Studio:

```
IPEConversionUtil.STRTOUPPER("test");
```

TIMESTR

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Convert a time to a string.

Syntax

```
timestr (time)
```

where *time* is the time to be converted.

Returns

A text string containing the text equivalent of the time passed, in the 24-hour format HH:MM (or otherwise according to your system configuration)

Chapter 5 Environment Functions

The following functions can be used to get and set environment information.

The ENQUIRE and ISWINDOWS functions are only of interest to users of the Staffware Application Layer (SAL) interface.

Function	Usage	Description	See
CUSTAUDIT		Add a user defined audit trail to a specific case's audit trail	page 55
ENQUIRE		Request information about environment	page 57
FORMCONTROL		Perform action on current form	page 58
FORMMAXIMIZE		Maximize the current form	page 60
FORMMINIMIZE		Minimize the current form	page 61
FORMMOVE		Move the current form	page 62
FORMRESTORE		Restore the current form	page 60
FORMSIZE		Change the size of the current form	page 64
GETHANDLE		Return handle	page 65
ISWINDOWS		Check if running TIBCO iProcess Workspace (Windows)	page 66
MARKFIELDCHANGED		Mark a field as changed	page 67
MEMOFILE		Return name of file containing text of a memo	page 68
MESSAGEBOX		Display message box	page 69
READFIELDS		Read values of fields from a file	page 71
SENDKEYS		Send keystrokes to the active window	page 73

Function	Usage	Description	See
SETSTEPSTATUS		Sets the status of a step to Not processed or Released	page 76
USERATTRIBUTE		Return a user's attribute value	page 79
WINACTION		Perform miscellaneous actions on a window	page 80
WINACTIVATE		Activate a window	page 82
WINCLOSE		Close a window	page 83
WINEXIST		Check if a window exists	page 84
WINFIND		Find a window to perform an action on	page 85
WINMAXIMIZE		Maximize the active window	page 87
WINMESSAGE		Display a message in window	page 88
WINMINIMIZE		Minimize the active window	page 90
WINMOVE		Move the active window	page 91
WINRESTORE		Restore the active window	page 93
WINSIZE		Change the size of the active window	page 94
WRITEFIELDS		Write current values of fields to a file	page 95

CUSTAUDIT

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Append user defined audit trail entries to a specified case's audit trail.

Syntax

```
custaudit(proc, casenum, auditID,
stepname, stepdesc, user)
```

where:

- *procname* is the name of the procedure that the case belongs to. If you specify *procname* as an empty string (""), it defaults to the current procedure at the current node.
- *casenum* is the case number to add the audit trail entry to. If you specify *casenum* as SW_CASENUM, it defaults to the current case.
- *auditID* is the audit trail entry ID, as defined in the *SWDIR***etc***language*.**lng****auditusr.mes** file. This must be a value between 256 and 999. (Values 0 to 255 are reserved for use by iProcess.)
- *stepname* is the step name. If you specify *stepname* as SW_STEPNAME, it defaults to the current step name. *stepname* must be 8 characters or less (unless you use SW_STEPNAME). Any characters above this are truncated.
- *stepdesc* is the step description. If you specify *stepdesc* as SW_STEPDESC, it defaults to the current step description. *stepdesc* must be 24 characters or less.
- *user* is the iProcess user. If you specify *user* as SW_USER:NAME, it defaults to the currently logged in iProcess user. *user* must be 255 characters or less.

You can provide any value for stepname, stepdesc and user but the interpretation of these values depends on the application used to display the audit trails. If you use iProcess audit trail windows, ensure there is an entry for the given audit trail ID in the SWDIR**etc**\language.**lng\auditusr.mes** file. The given values are used to replace the %USER and %DESC variables in the **auditusr.mes** message format string. Refer to "Audit Trails" in the *TIBCO iProcess swutil and swbatch Reference Guide* for more information about the **auditusr.mes** file.

Returns One of the following values:

Value	Description
0	Success.
1	Invalid auditID parameter.
2	Procedure or host name not found.
3	Case number not found.
4	CUSTAUDIT is not supported on this version of the iProcess Engine you are logged in to.
5	Failed to add the audit trail entry request to the queue.

Examples

TIBCO iProcess Modeler:

This example adds a user defined audit trail entry to **step1** of a procedure called **CARPOOL** for case number **52**.

```
custaudit (“carpool“, 52, 256, “step1“, “request for vehicle“,
“swusr001“)
```

This example adds a user-defined audit trail entry to the current step of the current procedure, for the current case.

```
custaudit ("", SW_CASENUM, 256, SW_STEPNAME, SW_STEPDESC,
SW_USER:NAME)
```

TIBCO Business Studio:

These examples are equivalent to the previous TIBCO iProcess Modeler examples.

```
IPEEnvironmentUtil.CUSTAUDIT("carpool", 52, 256, "step1", "request
for vehicle", "swusr001");

IPEEnvironmentUtil.CUSTAUDIT("", IPESystemValues.SW_CASENUM, 256,
IPESystemValues.SW_STEPNAME, IPESystemValues.SW_STEPDESC,
IPEStarterUtil.GETATTRIBUTE("Name"));
```

ENQUIRE

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Request information about environment.

Syntax

```
enquire (item)
```

where *item* is one of the following (not case sensitive):

Item (Text)	Returned value (Text)
OSName	Operating system where the TIBCO iProcess Objects (iPO) server is hosted, for example Windows or UNIX.
SAL Version	SAL version string where the iPO server is hosted.
FIL Version	iProcess FIL version string where the iPO server is hosted.
Server Version	TIBCO iProcess Engine version string.
Server OSName	Server operating system, for example Windows or UNIX.

Returns

The returned value of *item* as shown in the above table, or SW_NA if not recognized.

Examples

TIBCO iProcess Modeler:

This example displays the iProcess server version.

```
enquire(server version)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.ENQUIRE(server version);
```

FORMCONTROL

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Performs the specified action on the current form.

Syntax

```
formcontrol (action)
```

where *action* is one of the following numeric values, specifying the action to be done:

Value	Action
0	**abort** the form window; as a result it no longer exists and the form it represents must be kept or released by calling the appropriate SAL interface function
1	**undo** changes to field values since the form was opened
2	**keep** the form in the work queue after closing the window
3	**release** the form from the work queue after closing the window; this acts like **keep** if the form is not releasable
4	**hide** the form window so the user cannot interact with it; the window must subsequently be redisplayed with **show** or the form closed with **keep** or **release**
5	**show** the form window after **hide**

Returns One of the following numeric values:

Value	Description
0	Action not performed
1	Success

Examples

TIBCO iProcess Modeler:

`formcontrol (0)` aborts the form window so that all data entry can be performed in a separate application which calls the SAL API functions directly.

TIBCO Business Studio:

```
IPEEnvironmentUtil.FORMCONTROL(0);
```

FORMMAXIMIZE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

This function maximizes the current iProcess form (equivalent to the option on the form window's control menu).

Syntax

```
formmaximize ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Examples

TIBCO iProcess Modeler:

formmaximize()

TIBCO Business Studio:

```
IPEEnvironmentUtil.FORMMAXIMIZE();
```

FORMMINIMIZE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

This function minimizes the current iProcess form (equivalent to the option on the form window's control menu).

Syntax

```
formminimize ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Examples

TIBCO iProcess Modeler:

formminimize()

TIBCO Business Studio:

```
IPEEnvironmentUtil.FORMMINIMIZE();
```

FORMMOVE

Usage

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Moves the current iProcess form to the specified position on the screen.

Syntax

```
formmove (x, y)
```

where:

- *x* is a numeric value specifying:
  - if positive, the new horizontal position in points from the left edge of the screen.
  - if negative, the percentage across the screen width.
- *y* is a numeric value specifying:
  - if positive, the new vertical position in points from the top edge of the screen.
  - if negative, the percentage down the screen height.

Returns

One of the following numeric values:

Value	Description
0	Success
1	Failure

Examples

TIBCO iProcess Modeler:

```
formmove (20, 20)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.FORMMOVE(20,20);
```

FORMRESTORE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

This function restores the current iProcess form (equivalent to the option on the form window's control menu).

Syntax

```
formrestore ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Examples

TIBCO iProcess Modeler:

formrestore()

TIBCO Business Studio:

```
IPEEnvironmentUtil.FORMRESTORE();
```

FORMSIZE

Usage

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Changes the size of the current iProcess form.

Syntax

```
formsize (x, y)
```

where:

- *x* is a numeric value specifying:
  - — if positive, the new width of the form in points
  - — if negative, the new width as a percentage of the screen width.
- *y* is a numeric value specifying:
  - — if positive, the new height of the form in points
  - — if negative, the new height as a percentage of the screen height.

Returns

One of the following numeric values:

Value	Description
0	Success
1	Failure

Examples

TIBCO iProcess Modeler:

```
formsize (200, 200)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.FORMSIZE(200,200);
```

GETHANDLE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Returns the handle (internal reference number) of a specified item.

Syntax

```
gethandle (itemid)
```

where *itemid* is a numeric value specifying which item to return the handle of:

Value	Handle to return
0	SAL session handle
1	SAL mail session handle *(obsolete - always returns* **-1***).*
2	SAL form session handle
3	Form window handle
4	Work queue window handle
5	Tools window handle

Returns A numeric value which is the handle to be used in calls to SAL API functions. If the argument is invalid or the specified handle cannot be returned, the return value is **-1**.

Examples

TIBCO iProcess Modeler:

To return the SAL session handle in a call to a custom Windows application to handle form input:

```
winrun ("c:\myprog " + str (gethandle (0), 0), 1)
```

TIBCO Business Studio:

```
IPEExternalUtil.WINRUN("c:\\myprog " +
IPEConversionUtil.STR(IPEEnvironmentUtil.GETHANDLE(0),0),1);
```

ISWINDOWS

Usage

TIBCO iProcess Workspace (Windows)

Checks which version of the iProcess Workspace (Windows) the case is currently running under.

This function is superseded by the more general ENQUIRE function (see above), but is retained for upward compatibility.

Syntax

```
iswindows ()
```

Returns

One of the following Boolean values, depending on the platform:

Platform	Value	Description
UNIX	**TRUE**	Returned by client-based scripts executed by TIBCO iProcess Workspace (Windows).
UNIX	**FALSE**	• Returned by client-based scripts executed by TIBCO iProcess Workspace (Browser) - because the script will be executed by the SPO server running on UNIX. • Returned by all server-based scripts.
Windows	**TRUE**	All scripts return TRUE.

Examples

TIBCO iProcess Modeler:

```
iswindows()
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.ISWINDOWS();
```

MARKFIELDCHANGED

Usage

 TIBCO iProcess Workspace (Browser) - *discouraged* (triggers warning in TIBCO Business Studio)

 TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Mark a field as changed when a form is released.

Syntax

```
markfieldchanged (field, changed)
```

where:

- *field* is a text string specifying the name of the field.
- *changed* is one of the following numeric values:

Value	Field status
0	Mark field as unchanged
anything else	Mark field as changed

Returns

One of the following numeric values, indicating whether the field had been marked as changed before the function call.

Value	Description
0	Field was unchanged
-1	Invalid syntax
anything else	Field was changed

Examples

TIBCO iProcess Modeler:

```
markfieldchanged (dob, 1)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.MARKFIELDCHANGED(dob,1);
```

MEMOFILE

Usage

 TIBCO iProcess Workspace (Browser) - *discouraged* (triggers warning in TIBCO Business Studio)

 TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Returns the filename corresponding to the specified memo field.

Although memos are stored in the database, the MEMOFILE function extracts the memo from the database and stores it on the TIBCO iProcess Engine.

Syntax

```
memofile (memo)
```

where *memo* is a text string specifying the name of the memo field.

Returns

A text string containing the full pathname of the file which contains the text of the specified memo field.

If the memo field is SW_NA, the file will not yet exist but it may be created

Examples

TIBCO iProcess Modeler:

On iProcess Workspace (Windows), `memofile(comments)` could return:

```
d:\staff.dir\node.n\0301@a03.m01
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.MEMOFILE(comments);
```

MESSAGEBOX

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Displays a message dialog box.

Syntax

```
messagebox (title, message, icon, buttons)
```

where:

- *title* is a text string specifying the message box title.
- *message* is a text string specifying the message to show.
- *icon* is one of the following numeric values:

Value	Icon
0	no icon
1	!
2	I
3	?
4	Stop

- *buttons* is one of the following numeric values, specifying the available buttons:

Value	Button(s)
0	OK
1	OK/Cancel
2	Yes/No
3	Yes/No/Cancel

Value	Button(s)
4	Retry/Cancel

Returns One of the following numeric values:

Value	Description
-1	**Cancel** chosen
0	**No** chosen
1	**Yes**, **Retry** or **OK** chosen

Examples

TIBCO iProcess Modeler:

The following can be used to display a confirmation message:

```
messagebox ("confirm", "Are you sure you want to exit", 3, 2)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.MESSAGEBOX("confirm","Are you sure you want to
exit",3,2);
```

READFIELDS

Usage

TIBCO iProcess Workspace (Windows)

This function is also used with batch-oriented broker applications. For example, a TIBCO BusinessWorks service running as an iProcess broker via SSO use READFIELDS (in the step Initial script) on a file stored in *SWDIR*.

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Read values of selected fields from a file.

Syntax

```
readfields (filename, fieldlist, excluding, delete, srcfmt)
```

where:

- *filename* is a text string specifying the input file as a pathname or simple filename.

  The following environment variables may be used in a pathname:

  — TEMP, the temporary directory on the client.

  — HOME, the user's queue directory on the server, SWDIR**queues***username*.

  — SWDIR, the iProcess system directory where the TIBCO iProcess Engine is installed.

- *fieldlist* is a text string specifying the list of fields to be read, separated by commas. (Wildcard characters * and ? may be included.)
- *excluding* is a text string specifying the list of fields not to be read, even though selected in fieldlist.
- *delete* is a numeric value specifying which files to delete after reading data:

Value	File(s) to delete
0	Delete no files
1	Delete memo files
4	Delete input file

Value	File(s) to delete
5	Delete both memo files and input file

- *srcfmt* is a numeric value which is ignored in a standard installation (set to 0).

Returns

One of the following numeric values:

Value	Description
0	Error
1	Success

The input file is a text file in **abox** format, i.e. each line consists of a fieldname, followed by a comma, followed by the field value in characters. Variables in the input file must be in uppercase.

For a **memo** field, the value is the pathname of a text file containing the memo text; for an **attachment** field, the value is the pathname of the attachment file.

Examples

TIBCO iProcess Modeler:

This (Windows) example reads all field values from **abox** file DATA in the directory specified by the TEMP environment variable, deleting any memo files and the **abox** file after completion.

```
readfields("%TEMP%\DATA","*","",5,0)
```

If READFIELDS encounters any fields in the **abox** file which are not defined in the process, these fields are ignored and no errors are flagged (in particular, file *SWDIR***logs****sw_warn** is not updated).

TIBCO Business Studio:

```
IPEEnvironmentUtil.READFIELDS("%TEMP%\\DATA","*","",5,0);
```

SENDKEYS

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Sends the specified keystrokes to the active window. The active window may be an iProcess window, or another application window.

Syntax

```
sendkeys (keytext)
```

where *keytext* is a text string specifying the keys or key combinations to send. Any single key or any key combined with Alt, Ctrl, or Shift may be specified. The maximum number of keystrokes which may be represented by *keytext* is approximately 80:

- Printing keys are specified by the corresponding letter, for example **"a"**.
- Non-printing keys are specified by a key code included in *keytext*; for example **"{ENTER}"**. Valid codes are:

Key	Code
Backspace	"{BACKSPACE}", "{BS}" or "{BKSP}"
Break	"{BREAK}"
Caps Lock	"{CAPSLOCK}"
Clear	"{CLEAR}"
Delete	"{DELETE}" or "{DEL}"
Down Arrow	"{DOWN}"
End	"{END}"
Enter	"{ENTER}" or "~"
Esc	"{ESCAPE}" or "{ESC}"
Help	"{HELP}"

Key	Code
Home	"{HOME}"
Insert	"{INSERT}"
Left Arrow	"{LEFT}"
Page Down	"{PGDN}"
Page Up	"{PGUP}"
Print Screen	"{PRTSC}"
Right Arrow	"{RIGHT}"
Scroll Lock	"{SCROLLLOCK}"
Tab	"{TAB}"
Up Arrow	"{UP}"
F1 to F16	"{F1}" to "{F16}"

- Key combinations may be specified by preceding the key or keycode with one or more of the following characters:

To combine with	Use code
Shift	+
Ctrl	^
Alt	%

To use those characters or the bracket characters in their own right enclose them in brackets:

{+} {^} {%} {{} {}}

Invalid key sequences (for example, an unmatched `"{"`) results in no keys being sent.

Any key sequence directed to the Form Window which would result in the window being closed is intercepted. Use the ISWINDOWS function for this purpose. Similarly, attempting to close the **Tools** window with the SENDKEYS function should be avoided.

With some applications, it may be necessary to split a sequence of keys being sent into more than one SENDKEYS statement. This is to allow time for the application to respond.

Returns

One of the following numeric values:

Value	Description
0	Success
1	Failure (for example, too many keys in *keytext* argument, or unmatched brace)

Examples

TIBCO iProcess Modeler:

To send the keys **"Abc"** followed by **Enter**:

```
sendkeys("Abc{ENTER}")
```

To send the contents of the text field NAME:

```
sendkeys(name)
```

To send **Alt+E** followed by **C**; these keys normally select the application's Edit Copy facility:

```
sendkeys("%ec")
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.SENDKEYS("Abc{ENTER}");
IPEEnvironmentUtil.SENDKEYS("name");
IPEEnvironmentUtil.SENDKEYS("%ec");
```

SETSTEPSTATUS

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Sets the status of one or more steps in the current case to either **Not processed** or **Released**.

The SETSTEPSTATUS function returns a Boolean value, and so can be used in any iProcess condition expression. It would normally be used as a conditional action when a step is released. When the function is processed by the TIBCO iProcess Engine, the status of the specified steps is changed to the specified value.

The function:

- makes it easier to handle loop constructs which involve waits, by resetting step status before each iteration of the loop.
- allows the use of a wait to synchronize a number of concurrent paths in a procedure, some of which are exclusive, by setting the status of dependant steps on exclusive paths which are not travelled.

For more information about using SETSTEPSTATUS with waits, please see "Using SETSTEPSTATUS to Control the Loop" in the *TIBCO iProcess Modeler - Basic Design* guide.

Syntax

```
SETSTEPSTATUS (StepNameList, NewStatus)
```

where:

- *StepNameList* is a string which contains the names of all the steps whose status is to be changed. If more than one step name is supplied, the names should be separated by commas.
- *NewStatus* is one of the following numeric values:

Value	New step status
0	Not Processed
1	Released

Remarks SETSTEPSTATUS can only be used to set the status of steps which are either **Not Processed**, **Released** or **Withdrawn** when the function is processed. If a step is **Outstanding**, the function will return a FAIL value (see below).

Steps named in *StepNameList* are processed sequentially from left to right. A failure to set the requested status on a specific step does not prevent processing of the remainder of the steps in the list.

Returns One of the following Boolean values:

Value	Description
TRUE	if the specified status is successfully set for all specified steps
FALSE	otherwise

A **FALSE** return value will generate one or more of the following entries in the *SWDIR***Logs****Sw_warn** file.

SWDIR\logs\sw_warn Entry	Meaning
SetStepStatus - not processed *StepNameList*	The supplied *StepNameList* argument has an invalid value or is SW_NA.
SetStepStatus - not processed *NewStatus*	The supplied *NewStatus* argument has an invalid value or is SW_NA.
SetStepStatus - step *stepname* does not exist in procedure	The *stepname* supplied in *StepNameList* does not exist.
SetStepStatus - step *stepname* is outstanding, cannot be set to *status.*	The *stepname* supplied in *StepNameList* is **Outstanding**, so it could not be set to the indicated *status* (**Not Processed** or **Released**).
SetStepStatus - Failed to set the status of step *stepname* to *status.*	The *stepname* supplied in *StepNameList* could not be set to the indicated *status* (**Not Processed** or **Released**), because the SETSTEPSTATUS function is not supported on the executing platform (for example, on an earlier version of the iProcess Engine).

Examples

TIBCO iProcess Modeler:

1. This example sets the status of **step1** to **Not Processed**.

   ```
   SETSTEPSTATUS ("step1",0)
   ```

2. This example sets the status of **step1**, **step2** and **step3** to **Released**.

   ```
   SETSTEPSTATUS ("step1,step2,step3",1)
   ```

3. In this example, if **step1** was **Outstanding** when the function was processed by the server, the function would return a FAIL value. **step2** would still be set to **Released**.

   ```
   SETSTEPSTATUS ("step1,step2",1)
   ```

   Also, the following entry would be added to the *SWDIR***Logs****Sw_warn** file:

   ```
   SetStepStatus - step step1 is outstanding, cannot be set
   to Released.
   ```

TIBCO Business Studio:

```
IPEEnvironmentUtil.SETSTEPSTATUS("step1",0);
IPEEnvironmentUtil.SETSTEPSTATUS("step1,step2,step3",1);
IPEEnvironmentUtil.SETSTEPSTATUS("step1,step2",1);
```

USERATTRIBUTE

Usage

 TIBCO iProcess Workspace (Windows)

Return a user's or group's attribute value. For information about managing iProcess user attributes, see *TIBCO iProcess Workspace (Windows) Manager's Guide*.

Syntax

```
userattribute (user, attribute)
```

where:

- *user* is a text string specifying the name of the user or group.
- *attribute* is a text string specifying the name of the attribute.

Returns

A text string containing the attribute value of the user or group. Other possible return values include the following:

- If the user or group does not exist, it returns SW_NA.
- If the user or group does exist, but the accessed attribute does not exist, it returns SW_NA.
- If both user or group and its attribute exist, but there is no value for the attribute, it returns SW_BLANK.

Examples

TIBCO iProcess Modeler:

```
userattribute ("joseph","department")
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.USERATTRIBUTE("joseph","department");
```

WINACTION

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Perform miscellaneous actions on a window.

Syntax

```
winaction (handle, action, x, y)
```

where:

- *handle* is the numeric value, returned by the GETHANDLE function, indicating the window on which the action is to be performed.
- *action* is a number indicating the action to be performed on the window:

Value	Action
0	close window (supersedes WINCLOSE function)
1	activate window (supersedes WINACTIVATE function)
2	move window to coordinates *x*, *y* (supersedes WINMOVE function)
3	re-size window to width *x*, height *y* (supersedes WINSIZE function)
4	minimize window (supersedes WINMINIMIZE function)
5	maximize window (supersedes WINMAXIMIZE function)
6	restore window (supersedes WINRESTORE function)

- *x*, *y* are numeric values depend on action; ignored if irrelevant.

Returns One of the following numeric values:

Value	Description
0	Success

Value	Description
1	Failure

Example

TIBCO iProcess Modeler:

In a script:

```
MYNUMFLD := winfind ("Microsoft Excel", 3)
IF MYNUMFLD >= 0
; restore the window
winaction (MYNUMFLD, 6, 0, 0)
; activate the window
winaction (MYNUMFLD, 1, 0, 0)
; re-size the window
winaction (MYNUMFLD, 3, 300, 200)
ENDIF
```

TIBCO Business Studio:

The following closes the window:

```
MYNUMFLD=IPEEnvironmentUtil.WINFIND("Microsoft Excel",3);
if(MYNUMFLD >= 0) {
IPEEnvironmentUtil.WINACTION(MYNUMFLD,0);
}
```

WINACTIVATE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Makes the specified window active. This function does not affect whether the specified window is maximized or minimized.

This function is superseded by the more general WINACTION function, but is retained for upward compatibility.

Syntax

```
winactivate (title)
```

where *title* is a text string specifying all or the first part of the title bar of the application window to activate. If there is more than one matching window, the one to be activated will be arbitrarily selected. Matching of the title is case insensitive.

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

```
winactivate ("Microsoft Excel")
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINACTIVATE("Microsoft Excel");
```

WINCLOSE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Closes the specified window. You should avoid using this function to close the Tools Window, otherwise there may be undesirable results.

This function is superseded by the more general WINACTION function, but is retained for upward compatibility.

This function cannot be used to close the Form Window. Use the ISWINDOWS function for this purpose.

Syntax

```
winclose (title)
```

where *title* is a text string specifying all or the first part of the title bar of the application window to close. If there is more than one matching window, the one to be closed will be arbitrarily selected. Matching of the title is case insensitive.

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

```
winclose ("Microsoft Excel")
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINCLOSE("Microsoft Excel");
```

WINEXIST

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Checks if the specified window exists.

Syntax

```
winexist (title)
```

where *title* is a text string which specifies all or the first part of the title bar of the application window whose existence is to be checked for. Matching of the title is case insensitive.

Returns One of the following Boolean values:

Value	Description
TRUE	The window exists.
FALSE	The window does not exist.

Example

TIBCO iProcess Modeler:

In a script:

```
IF winexist ("Microsoft Excel")
    winactivate ("Microsoft Excel")
ELSE
    winrun ("C:\EXCEL\EXCEL", 1)
ENDIF
```

TIBCO Business Studio:

```
if(IPEEnvironmentUtil.WINEXIST("Microsoft Excel"))  {
    IPEEnvironmentUtil.WINACTIVATE("Microsoft Excel");
} else {
        IPEExternalUtil.WINRUN("C:\\EXCEL\\EXCEL", 1);
}
```

WINFIND

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Find a window to perform an action on.

Syntax

```
winfind (title, modifier)
```

where:

- *title* is a text string which specifies all or the first part of the title bar of the application window to be checked. Matching of the title is case insensitive.
- *modifier* is a numeric value which indicates what windows should be included in the search:

Value	Windows to search
0	All top level and child windows
1	All *visible* top level and child windows (i.e. not hidden)
2	All top level windows
3	All *visible* top level windows (i.e. not hidden)

Returns (numeric) The 'handle' of the window found. This value should be used when calling the WINACTION function. If a matching window cannot be found, or the modifier is invalid, the return value is 0.

Example

TIBCO iProcess Modeler:

In a script:

```
MYNUMFLD := winfind ("Microsoft Excel", 3)
IF MYNUMFLD >= 0
messagebox ("Found Microsoft Excel.", "Window handle = " + str
(MYNUMFLD, 0), 0, 0)
ENDIF
```

TIBCO Business Studio:

```
MYNUMFLD=IPEEnvironmentUtil.WINFIND("Microsoft Excel",3);
if(MYNUMFLD >= 0) {
IPEEnvironmentUtil.MESSAGEBOX("Found Microsoft Excel.","Window
handle = " + IPEConversionUtil.STR(MYNUMFLD, 0), 0, 0);
}
```

WINMAXIMIZE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Maximizes the active window (equivalent to the option on the window's control menu).

This function has been superseded by the more general WINACTION function, but is retained for upward compatibility.

Syntax

```
winmaximize ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

In a script:

```
winactivate ("Microsoft Excel")
winmaximize ()
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINACTIVATE("Microsoft Excel");
IPEEnvironmentUtil.WINMAXIMIZE();
```

WINMESSAGE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Display a message in a small window. The window appears above all others and the text is shown in up to 3 lines of about 40 characters each; the message is word-wrapped, or you may force a new line with a **\n** sequence (see the example below).

Syntax

```
winmessage (mesg, x, y)
```

where:

- *mesg* is a text string specifying the message to display in the window. The window is created if it does not exist; otherwise the existing one is used. A call to the function with the null string **""** as a message closes the window.
- *x* is one of the following numeric values, specifying the horizontal position of the window on the screen:

Value	Window's horizontal position
0	Left side
1	Center
2	Right side

- *y* is one of the following numeric values, specifying the vertical position of the window on the screen:

Value	Window's vertical position
0	Top
1	Center
2	Bottom

Returns One of the following Boolean values:

Value	Description
TRUE	Success
FALSE	Failure

Example

TIBCO iProcess Modeler:

In a script:

```
winmessage("Scanning\nPlease wait", 1, 1)
; Show message (on two lines) in a window
; in the center of the screen during
; iProcess function calls.
winmessage ("", 0, 0) ; remove message box
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINMESSAGE("Scanning\nPlease wait", 1, 1);
/* Show message (on two lines) in a window
 * in the center of the screen during
 * iProcess function calls. */
IPEEnvironmentUtil.WINMESSAGE("", 0, 0); // remove message box
```

WINMINIMIZE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Minimizes the active window (equivalent to the option on the window's control menu).

This function has been superseded by the more general WINACTION function, but is retained for upward compatibility.

Syntax

```
winminimize ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

In a script:

```
winactivate ("Microsoft Excel")
winminimize ()
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINACTIVATE("Microsoft Excel");
IPEEnvironmentUtil.WINMINIMIZE();
```

WINMOVE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Moves the active window to the specified position on the screen. The active window may be an iProcess window, or another application window.

This function is superseded by the more general WINACTION function, but is retained for upward compatibility.

Syntax

```
winmove (x, y)
```

where:

- *x* is a numeric value specifying:
  - if positive, the new horizontal position in points from the left edge of the screen.
  - if negative, the percentage across the screen width.
- *y* is a numeric value specifying:
  - if positive, the new vertical position in points from the top edge of the screen.
  - if negative, the percentage down the screen height.

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

```
winmove (20, 20)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINMOVE(20,20);
```

WINRESTORE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Restores the active window (equivalent to the option on the window's control menu).

This function has been superseded by the more general WINACTION function, but is retained for upward compatibility.

Syntax

```
winrestore ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

In a script:

```
winactivate ("Microsoft Excel")
winminimize ()
winrestore ()
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINACTIVATE("Microsoft Excel");
IPEEnvironmentUtil.WINMINIMIZE();
IPEEnvironmentUtil.WINRESTORE();
```

WINSIZE

Usage

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Changes the size of the active window. The active window may be an iProcess window, or another application window.

This function is superseded by the more general WINACTION function, but is retained for upward compatibility.

Syntax

```
winsize (x, y)
```

where:

- *x* is a numeric value specifying:
  - — if positive, the new width of the window in points.
  - — if negative, the new width as a percentage of the screen width.
- *y* is a numeric value specifying:
  - — if positive, the new height of the window in points.
  - — if negative, the new height as a percentage of the screen height.

Returns

One of the following numeric values:

Value	Description
0	Success
1	Failure

Example

TIBCO iProcess Modeler:

```
winsize (200, 200)
```

TIBCO Business Studio:

```
IPEEnvironmentUtil.WINSIZE(200,200);
```

WRITEFIELDS

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Write current values of selected fields to a file.

Syntax

```
writefields (filename, fieldlist, excluding, copy, destfmt)
```

where:

- *filename* is a text string specifying the output file as either a pathname, a simple filename or the full path of the directory for the file, in which case a unique filename will be generated (and returned by the function).

  The following environment variables may be used in a pathname:

  — TEMP, the temporary directory on the client.

  — HOME, the user's queue directory on the server, SWDIR**queues***username*.

This directory is not created automatically. This means that before using this environment variable you should check that it exists, otherwise the function will fail.

- *fieldlist* is a text string specifying the list of fields to be written, separated by commas. (Wildcard characters * and ? may be included.)
- *excluding* is a text string specifying the list of fields not to be written, even though selected in *fieldlist*.
- *copy* is one of the following numeric values, specifying whether to make copies of memos and/or attachment files:

Value	Files to copy
0	Copy neither
1	Copy memos
2	Copy attachments

Value	Files to copy
3	Copy both memos and attachments

Memos and attachment files are copied to the same directory as the output file under their original names, and their new pathnames written to the output file as the values of the fields. If they are not copied, their original pathnames are written to the output file. (If not copied the files may not be available to the client, or may be deleted or changed.)

- *destfmt* is one of the following numeric values, specifying the destination format of the output file *filename*:

Value	Destination format
0	Current client
1	UNIX
3	DOS type

This affects line-terminator characters.

Returns A text string containing the full pathname of the output file, or SW_NA on error.

The output file is a text file in **abox** format - each line consists of the fieldname, followed by a comma, followed by the field value in characters. Passwords are not written.

Example

TIBCO iProcess Modeler:

This (Windows) example writes the contents of all fields excluding (SW_*) system fields with DOS line terminators to a computer-generated filename in the directory specified by the TEMP environment variable, and puts the full path of the output file in text field **abox**. Any memos and attachment files will be copied to the same directory and their pathnames written to the output file.

```
abox := writefields("%TEMP%","*","SW_*",3,3)
```

TIBCO Business Studio:

```
abox = IPEEnvironmentUtil.WRITEFIELDS("%TEMP%","*","SW_*",3,3);
```

Chapter 6 File Functions

The following functions can be used to manipulate files.

Function	Usage	Description	See
FILECOPY		Copy a file	page 98
FILEDELETE		Delete a file	page 99
FILEEXISTS		Check if a file exists	page 100
FILERENAME		Rename a file	page 101
FILEREQUEST		Request file selection	page 103

FILECOPY

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Copy a file.

Syntax

```
filecopy (source, dest)
```

where:

- *source* is a text string specifying the name of the file to be copied (which must exist).
- *dest* is a text string specifying the filename to be copied to (which will be overwritten of it already exists), as either a simple filename or full pathname.

  The following environment variables may be used in a pathname:

  — TEMP, the temporary directory on the iProcess Workspace client.

  — HOME, the user's queue directory on the server, SWDIR**queues***username*.

This directory is not created automatically. This means that before using this environment variable you should check that it exists, otherwise the function will fail.

Returns

One of the following numeric values:

Value	Description
1	Success
-4	Failed to open either file
-2	Failed to copy for any other reason.

Examples

TIBCO iProcess Modeler:

This command makes a backup copy of a log file in the temporary directory.

```
filecopy ("%TEMP%\log.txt","%TEMP%\log_backup.txt")
```

TIBCO Business Studio:

```
IPEFileUtil.FILECOPY("%TEMP%\\log.txt","%TEMP%\\log_backup.txt");
```

FILEDELETE

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Delete a file.

Syntax

```
filedelete (filename)
```

where *filename* is a text string specifying the name of the file to be deleted, as either a simple filename or full path.

The following environment variables may be used in a pathname:

- TEMP, the temporary directory on the iProcess Workspace client.
- HOME, the user's queue directory on the server, SWDIR**queues***username*.

This directory is not created automatically. This means that before using this environment variable you should check that it exists, otherwise the function will fail.

Returns

One of the following numeric values:

Value	Description
1	Success
-4	Failed to delete file for any reason.

Examples

TIBCO iProcess Modeler:

This command deletes a file in the current directory.

```
filedelete ("names.txt")
```

TIBCO Business Studio:

```
IPEFileUtil.FILEDELETE("names.txt");
```

FILEEXISTS

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Check if a file exists.

Syntax

```
fileexists (filename)
```

where *filename* is a text string specifying the name of the file to be checked, as either a simple filename or full pathname.

The following environment variables may be used in a pathname:

- TEMP, the temporary directory on the iProcess Workspace client.
- HOME, the user's queue directory on the server, SWDIR**queues***username*.

This directory is not created automatically. This means that before using this environment variable you should check that it exists, otherwise the function will fail.

Returns

One of the following Boolean values:

Value	Description
TRUE	File exists
FALSE	File does not exist

Examples

TIBCO iProcess Modeler:

This command checks to see if a log file exists in the temporary directory.

```
fileexists ("%TEMP%\log.txt")
```

TIBCO Business Studio:

```
IPEFileUtil.FILEEXISTS("%TEMP%\\log.txt");
```

FILERENAME

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Rename a file.

Syntax

```
filerename (source, dest)
```

where:

- *source* is a text string specifying the name of the file to be renamed (which must exist).
- *dest* is a text string specifying the new filename (which will be overwritten of it already exists) as a simple filename or full pathname.

  The following environment variables may be used in a pathname:

  — TEMP, the temporary directory on the iProcess Workspace client.

  — HOME, the user's queue directory on the server, SWDIR**queues***username*.

This directory is not created automatically. This means that before using this environment variable you should check that it exists, otherwise the function will fail.

You may rename a file to be in another directory, provided that directory exists. If you rename it to be on a different physical device, it is first copied, then the original deleted.

Returns

One of the following numeric values:

Value	Description
1	Success
-4	Failed to open either file.

Examples

TIBCO iProcess Modeler:

This command renames a file in the current directory.

```
filerename ("names.txt","names_old.txt")
```

TIBCO Business Studio:

```
IPEFileUtil.FILERENAME("name.txt","name_old.txt");
```

FILEREQUEST

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Requests file selection from the user using the common dialog.

Syntax `filerequest (`*`title`*`, `*`initdir`*`, `*`filters`*`, `*`initpath`*`)`

where:

- *title* is a text string specifying the dialog title.
- *initdir* is a text string specifying the initial directory.
- *filters* is a text string specifying the filters, in the form:

  description;filter;description;filter...

  SW_NA or "" means *.*.

  For example:

  `"Text Files;*.TXT"`.
- *initpath* is a text string specifying the initial path and/or filename.

Returns A text string containing the pathname of the file selected.

Examples

TIBCO iProcess Modeler:

This prompts the user to choose text files from the specified directory in the **Choose a File** dialog.

```
filerequest ("Choose a File","C:\", "Text
Files,*.TXT","C:\names.txt")
```

TIBCO Business Studio:

```
IPEFileUtil.FILEREQUEST("Choose a File","C:\\", "Text
Files,*.TXT","C:\\names.txt");
```

Chapter 7 Date and Time Functions

The following functions can be used to get and set date and time data.

Function	Usage	Description	See
CALCDATE		Add days, weeks, months and years to a date	page 106
CALCTIME		Add hours and minutes to a time	page 107
DATE		Construct date from day, month and year	page 108
DAYNUM		Return day number of a date	page 109
DAYSTR		Return day name of a date	page 109
HOURNUM		Return hours part of a time	page 111
MINSNUM		Return minutes part of a time	page 111
MONTHNUM		Return month number of a date	page 109
MONTHSTR		Return month name of a date	page 109
TIME		Construct time from hour and minute	page 115
WEEKNUM		Return week number of a date	page 109
YEARNUM		Return years part of a date	page 109

CALCDATE

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Performs a calculation on a date and returns a new date.

Syntax

```
calcdate (datein, days, weeks, months, years)
```

where:

- *datein* is the date the calculation is to be performed on and *the date* is not prior to 01/01/1900 (calcdate will not calculate dates correctly when *datein* is prior to 01/01/1900).
- *days, weeks, months* and *years* are numeric values (positive or negative) which are added to *datein*.

The calculation is performed according to the current working days configuration.

Returns

date

Examples

TIBCO iProcess Modeler:

`calcdate(SW_DATE, 0, 0, 1, 0)` returns today's date incremented by a month.

TIBCO Business Studio:

```
IPEDateTimeUtil.CALCDATE(IPESystemValues.SW_DATE, 0, 0, 1, 0);
```

CALCTIME

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Performs a calculation on a time and returns a new time.

Syntax

`calctime (timein, hours, minutes, daysover)`

where:

- *timein* is the time the calculation is to be performed on.
- *hours* and *minutes* are numeric values (positive or negative) which are added to *timein*.
- *daysover* is the **name** of a numeric field which returns with the number of days overflowed from the calculation.

The calculation is performed according to the current working days configuration.

Returns

time

Examples

TIBCO iProcess Modeler:

```
calctime (#12:00#, 2, 40, daysover)
```

returns `14:40` (daysover field = 0).

```
calctime (#13:35#, 12, 0, daysover)
```

returns `01:35` (daysover field = 1).

TIBCO Business Studio:

```
IPEDateTimeUtil.CALCTIME("12:00",2,40,daysover);
IPEDateTimeUtil.CALCTIME("13:35",12,0,daysover);
```

DATE

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Constructs a date from the specified day, month and year

Syntax

`date (`*`day`*`, `*`month`*`, `*`year`*`)`

where:

- *day* is a numeric value in the range **1-***(number of days in month)*
- *month* is a numeric value in the range **1-12**
- *year* is a numeric value in the range **0-2999**

Returns

date (or SW_NA for an invalid date)

Examples

TIBCO iProcess Modeler:

This example returns the date of the first day of the next month:

```
date (1, monthnum (calcdate (sw_date, 0, 0, 1,
0)), yearnum (calcdate (sw_date, 0, 0, 1, 0)))
```

TIBCO Business Studio:

```
IPEDateTimeUtil.DATE(1,
IPEDateTimeUtil.MONTHNUM(IPEDateTimeUtil.CALCDATE(IPESystemValues.
SW_DATE, 0, 0, 1, 0)),
IPEDateTimeUtil.YEARNUM(IPEDateTimeUtil.CALCDATE(IPESystemValues.
SW_DATE, 0, 0, 1, 0)));
```

DAYNUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the day number (in the month) of a specified date.

Syntax

`daynum (`*`date`*`)`

where *date* is the date the operation is to be performed on.

Returns

A numeric value containing the day number in the month of the *date*.

Examples

TIBCO iProcess Modeler:

```
DAYNUM (!08/10/2001!)
```

returns 8.

TIBCO Business Studio:

```
IPEDateTimeUtil.DAYNUM("08/10/2001");
```

DAYSTR

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the day name of a specified date.

Syntax

`daystr (`*`date`*`)`

where *date* is the date the operation is to be performed on.

Returns

A text string representing the day name of the *date*.

Examples

TIBCO iProcess Modeler:

```
DAYSTR (!08/10/2001!)
```

returns `"Monday"`.

TIBCO Business Studio:

```
IPEDateTimeUtil.DAYSTR("08/10/2001");
```

HOURNUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the minutes component of a specified time.

Syntax

`hournum (`*`time`*`)`

where *time* is the time the operation is to be performed on.

Returns

A numeric value representing the minutes component (0 to 23) of the *time*.

Examples

TIBCO iProcess Modeler:

```
HOURNUM (#06:24#)
```

returns `6`.

TIBCO Business Studio:

```
IPEDateTimeUtil.HOURNUM("06:24");
```

MINSNUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the minutes component of a specified time.

Syntax

`minsnum (`*`time`*`)`

where *time* is the time the operation is to be performed on.

Returns

A numeric value representing the minutes component (0 to 59) of the *time.*

Examples

TIBCO iProcess Modeler:

```
MINSNUM (#06:24#)
```

returns `24`.

TIBCO Business Studio:

```
IPEDateTimeUtil.MINSNUM("06:24");
```

MONTHNUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the month number (in the year) of a specified date.

Syntax

`monthnum (`*`date`*`)`

where *date* is the date the operation is to be performed on.

Returns

A numeric value containing the month number in the year (1 to 12) of the *date*.

Examples

TIBCO iProcess Modeler:

```
MONTHNUM (!08/10/2001!)
```

returns `10`.

TIBCO Business Studio:

```
IPEDateTimeUtil.MONTHNUM("08/10/2001");
```

MONTHSTR

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the month name of a specified date.

Syntax

`monthstr (`*`date`*`)`

where *date* is the date the operation is to be performed on.

Returns

A text string representing the month name of the *date*.

Examples

TIBCO iProcess Modeler:

```
MONTHSTR (!08/10/2001!)
```

returns `"October"`.

TIBCO Business Studio:

```
IPEDateTimeUtil.MONTHSTR("08/10/2001");
```

TIME

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Constructs a time from the specified hours and minutes.

Syntax

`time (`*`hours`*`, `*`minutes`*`)`

where:

- *hours* is a numeric value in the range **0-23**
- *minutes* is a numeric value in the range **0-59**

Returns

The resulting time.

Examples

TIBCO iProcess Modeler:

```
TIME (6,24)
```

returns `06:24`.

TIBCO Business Studio:

```
IPEDateTimeUtil.TIME(6,24);
```

WEEKNUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the week number (in the year) of a specified date.

Syntax

`weeknum (`*`date`*`)`

where *date* is the date the operation is to be performed on.

Returns

A numeric value containing the week number in the year (1 to 52 or 53, as appropriate) of the *date*.

Examples

TIBCO iProcess Modeler:

```
WEEKNUM (!08/10/2001!)
```

returns `41`.

TIBCO Business Studio:

```
IPEDateTimeUtil.WEEKNUM("08/10/2001");
```

YEARNUM

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the year number of a specified date.

Syntax

`yearnum (`*`date`*`)`

where *date* is the date the operation is to be performed on.

Returns

A numeric value containing the year number (0 to 2999) of the *date*.

Examples

TIBCO iProcess Modeler:

```
YEARNUM (!08/10/2001!)
```

returns `2001`.

TIBCO Business Studio:

```
IPEDateTimeUtil.YEARNUM("08/10/2001");
```

Chapter 8 String (Text) Functions

The following functions can be used to manipulate text strings.

Function	Usage	Description	See
RSEARCH		Search for a string in another string (back)	page 120
SEARCH		Search for a string in another string	page 121
STRLEN		Return number of characters in a string	page 122
SUBSTR		Return part of a string	page 123

RSEARCH

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Searches backwards for a string in another string.

Syntax

```
rsearch (search, target)
```

where:

- *search* is the text string to search for.
- *target* is the text string to be searched.

Returns

One of the following numeric values:

Value	Description
0	No match
>0	A match was found. The value indicates the character position of the start of the *search* string from the start of the *target* string (counting from 1).

Examples

TIBCO iProcess Modeler:

```
rsearch("abc", "junkabcdefs")
```

returns the value `5`.

```
rsearch("abc", "a")
```

returns the value `0`.

TIBCO Business Studio:

```
IPEStringUtil.RSEARCH("abc", "junkabcdefs");
IPEStringUtil.RSEARCH("abc", "a");
```

SEARCH

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Searches for a string in another string.

Syntax

```
search (search, target)
```

where:

- *search* is the text string to search for.
- *target* is the text string to be searched.

Returns

One of the following numeric values:

Value	Description
0	No match
>0	A match was found. The value indicates the character position of the start of the *search* string from the start of the *target* string (counting from 1).

Examples

TIBCO iProcess Modeler:

```
search("abc", "junkabcdefs")
```

returns the value `5`.

```
search("abc", "a")
```

returns the value `0`.

TIBCO Business Studio:

```
IPEStringUtil.SEARCH("abc", "junkabcdefs");
IPEStringUtil.SEARCH("abc", "a");
```

STRLEN

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Returns the number of characters in a string.

Syntax

```
strlen (text)
```

where *text* is a text string.

Returns

The length in characters of *text*.

Examples

TIBCO iProcess Modeler:

```
strlen("")
```

returns the value `0`.

```
strlen("abcdef")
```

returns the value `6`.

TIBCO Business Studio:

```
IPEStringUtil.STRLEN("");
IPEStringUtil.STRLEN("abcdef");
```

SUBSTR

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

Return part of a string.

Syntax

```
substr (text, start, length)
```

where:

- *text* is the text string to be operated upon.
- *start* is a numeric value specifying the character position in *text* at which to start (counting from 1).
- *length* is a numeric value specifying the number of characters to extract from *text*, starting from *start*.

Returns

The modified *text* string.

Examples

TIBCO iProcess Modeler:

```
substr("abcdefgh", 3, 3)
```

returns the value `"cde"`.

```
substr("abcdefgh", 10, 1)
```

returns the value `""`.

TIBCO Business Studio:

```
IPEStringUtil.SUBSTR("abcdefgh", 3, 3);
IPEStringUtil.SUBSTR("abcdefgh", 10, 1);
```

Chapter 9 Functions to Call External Programs

The following functions can be used to call external programs on the iProcess Engine or TIBCO iProcess Workspace (Windows).

Function	Usage	Description	See
SERVEREXEC		Run a Server program (no shell)	page 130
SERVERRUN		Run a Server program	page 130
UNIXEXEC		Run a Server program (no shell)	page 130
UNIXRUN		Run a Server program	page 130
WINRUN		Start program on iProcess Workspace (Windows)	page 133

SERVEREXEC

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Run a program on the server machine on behalf of the current user.

In addition to running the server program, any **abox** file that is present is processed.

SERVEREXEC and SERVERRUN work identically, except that SERVEREXEC does not start up a program shell; this should be more efficient. However, make sure that the shell program specifier, for example **#!/bin/sh** is added to the script so that the operating system treats the script as a shell script rather than a text file. If the shell program specifier is not added, this can cause the script to fail on some operating systems.

The SERVERRUN, UNIXEXEC and UNIXRUN functions are retained to ensure compatibility with earlier applications.

Syntax

```
serverexec (cmdline, async)
```

where:

- *cmdline* is a text string specifying the command line to run the server program (including any parameters).

Don't forget the extension for programs in DOS/Windows.

The command line may be defined as any other text expression, for example:

```
serverexec("\bin\myprog param1", 0)
```

or

```
serverexec("\bin\myprog " + text, 0)
```

Maximum length of the command line is 255 characters.

Alternatively, fields may be specified by enclosing them in **ampersands** (“`&...&`”), for example:

```
serverexec("\bin\myprog &text&", 0)
```

In this case, the current value of field TEXT will be substituted for **&text&**. If the field is SW_NA, the value passed is a hyphen surrounded by single quotes (**'-'**). If the field does not exist, the parameter is interpreted literally.

The maximum length of the command line using ampersands is 990 characters (after expanding fields).

Whichever method is used to specify a field, if the fieldname refers to a memo field, the program receives the full pathname of a file containing the memo text. If it refers to an attachment field, the program receives the full path of the attachment file.

You must use the full pathname with SERVEREXC. If you do not, SERVEREXEC will return a **-5** error.

- *async* is a numeric value specifying whether the server program should be run asynchronously or synchronously.

Value	Program behavior
non-0	iProcess does not wait for the program to exit before continuing.
0	iProcess does wait for the program to exit before continuing. In this case, after the server program has exited, the **abox** file in the *SWDIR***queues***username* directory (if present), is processed.

If the SERVEREXEC function is evaluated on iProcess Workspace (Windows), it is treated as if the value of *async* is always **0**.

It is not possible to invoke an interactive server program using SERVEREXEC on iProcess Workspace (Windows).

'abox' File Processing

This section applies to Field or Form Commands only.

This file is created by the external program to pass data, and certain special instructions, back to iProcess. It must be called **abox** and be located in the *SWDIR***queues***username* directory on return from the external program. (This directory will be current when the program is called.)

To pass data back to iProcess the file should contain one or more ASCII text lines consisting of the name of the field (in capitals), followed by a comma, followed by the data, for example:

```
CUST_NAME,William Hoycliffe
CUST_BALANCE,153.54
```

If there is no text after the comma, the field is set to SW_NA.

Text data is NOT enclosed in quote marks; also delimiters are NOT used for date or time data - just enter the figures with the appropriate separators, i.e. DD/MM/YYYY for dates and HH:MM for times.

Returns One of the following numeric values:

- On a UNIX server:

Value	Description
-6	*cmdline* too long to run the program
-5	Could not write request to SERVEREXEC daemon
-4	*cmdline* is blank or SW_NA
-3	Timed-out waiting for SERVEREXEC daemon response. The time out period is defined in the SWDIR**etc****staffcfg** configuration file on the server by the item FGLITO.
-2	Could not execute server program
-1	Problem during execution of server program
Greater than 0	Server program's exit code

- On a Windows server:

Value	Description
-5	Could not write request to SERVEREXEC daemon
-2	Could not execute server program
-1	Problem during execution of server program. **Note: When SERVEREXEC is run on iProcess Workspace (Windows), 1 is returned if the program cannot be executed, not -2 or -1.**
Other	Server program's exit code

Examples

TIBCO iProcess Modeler:

The following examples are all equivalent (although the third example is only possible as a Field or Form Command):

```
serverexec("\usr\bin\dbupdate CUST &fld1&&fld2&", 0)
serverexec("\usr\bin\dbupdate CUST " + fld1 + fld2, 0)
dbupdate CUST &fld1&&fld2&
```

TIBCO Business Studio:

```
IPEExternalUtil.SERVEREXEC("\\usr\\bin\\dbupdate CUST
&fld1&&fld2&", 0);
IPEExternalUtil.SERVEREXEC("\\usr\\bin\\dbupdate  CUST " + fld1 +
fld2, 0);
```

SERVERRUN

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Run a program on the server machine on behalf of the current user.

In addition to running the server program, any **abox** file that is present is processed.

SERVEREXEC and SERVERRUN work identically, except that SERVEREXEC does not start up a program shell; this should be more efficient. The SERVERRUN, UNIXEXEC and UNIXRUN functions are retained to ensure compatibility with earlier applications.

Syntax

`serverrun (cmdline, async)`

See the description of SERVEREXEC for more information about the *cmdline* and async *parameters*.

Returns

The same values as for SERVEREXEC.

UNIXEXEC

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Run a program on the server machine on behalf of the current user.

In addition to running the server program, any **abox** file that is present is processed.

UNIXEXEC and UNIXRUN work identically, except that UNIXEXEC does not start up a program shell; this should be more efficient. However, make sure that the shell program specifier, for example **#!/bin/sh** is added to the script so that the operating system treats the script as a shell script rather than a text file. If the shell program specifier is not added, this can cause the script to fail on some operating systems.

The SERVERRUN, UNIXEXEC and UNIXRUN functions are retained to ensure compatibility with earlier applications.

Syntax

```
unixexec (cmdline, async)
```

See the description of SERVEREXEC for more information about the *cmdline* and async *parameters*.

Returns

The same values as for SERVEREXEC

UNIXRUN

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Run a program on the server machine on behalf of the current user.

In addition to running the server program, any **abox** file that is present is processed.

UNIXEXEC and UNIXRUN work identically, except that SERVEREXEC does not start up a program shell; this should be more efficient. The SERVERRUN, UNIXEXEC and UNIXRUN functions are retained to ensure compatibility with earlier applications.

Syntax

`unixrun(cmdline, async)`

See the description of SERVEREXEC for more information about the *cmdline* and async *parameters*.

Returns

The same values as for SERVEREXEC

WINRUN

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Start a program on iProcess Workspace (Windows).

This function differs from SERVERRUN in that iProcess never waits for the program to exit before continuing, and that no **abox** file processing is performed.

Syntax `winrun (cmdline, show)`

where:

- *cmdline* is a text string specifying the command line which would be used to start the application; this can contain parameters specified in the same way as for SERVERRUN above.
- *show* is one of the following numeric values, specifying how the application window is shown initially, unless overridden by the application.

Value	Initial window view
0, **1**, or **>5**	Show the window normal size, and activate it
2	Show the window minimized, and activate it
3	Show the window maximized, and activate it
4	Show the window normal size, do not activate it
5	Show the window minimized, do not activate it

Returns One of the following numeric values:

Value	Description
-6	*cmdline* too long to run (maximum 127 characters).
-1	*cmdline* is blank or SW_NA.

Value	Description
0 - 31	iProcess Workspace (Windows) error code.
>= 32	Success (instance handle returned – an integer of up to 6 digits).

Examples

TIBCO iProcess Modeler:

```
winrun ("EXCEL " + datafile + ".XLS", 1)
```

TIBCO Business Studio:

```
IPEExternalUtil.WINRUN("EXCEL " + datafile + ".XLS", 1);
```

Chapter 10 Validation Functions

The following functions can be used to add data from a file to a field's validation list.

Function	Usage	Description	See
VLDFILE		Add data from a file to validations list	page 136
VLDFILEX		Add data from a file to validations list (extended)	page 136
VLDQUERY		Add data from database to validations list	page 140

VLDFILE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Adds each line from a text file to the list of validations to be used on the current field. Only relevant when in the validations **Values** column; ignored elsewhere.

Syntax

```
vldfile (file, location, limit)
```

where:

- *file* is a text string specifying the name of the text file containing the validations; either the full pathname, or a simple filename in the directory specified by the location parameter. (In the latter case the filename will be case sensitive.)

A simple filename may be a maximum of 12 characters, and a full pathname 255 characters.

- *location* is a numeric value specifying the directory where the file is located IF *file* is a simple filename:

Value	Directory
0	Central *SWDIR***lists** directory
1	User's *SWDIR***queues***username* directory

- *limit* is a numeric value specifying the maximum number of validations to add to the validations list (subject to the overall limit specified by the MAXVLD entry in the *SWDIR***etc****staffcfg** file, or 1000 if there is no such entry).

Returns The number of validations added to the validations list.

The list is created when the form is first displayed, and cannot be regenerated while the form is open.

Examples

TIBCO iProcess Modeler:

File **partnums** in the *SWDIR***lists** directory:

```
vldfile ("partnums", 0, 50)
```

Simple filename contained in text field LISTFLD, file in the user's *SWDIR***queues***username* directory:

```
vldfile (LISTFLD, 1, 10)
```

The specified file on the user's machine:

```
vldfile ("C:\DATA\LIST.TXT", 0, 100)
```

TIBCO Business Studio:

```
IPEValidationUtil.VLDFILE("partnums", 0, 50);
IPEValidationUtil.VLDFILE(LISTFLD, 1, 10);
IPEValidationUtil.VLDFILE("C:\\DATA\\LIST.TXT", 0, 100);
```

VLDFILEX

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Adds each line from a text file to the list of validations to be used on the current field. Only relevant when in the validations **Values** column; ignored elsewhere.

Syntax `vldfilex (file, location, limit, format, charset)`

where:

- *file* is a text string specifying the name of the text file containing the validations; either the full pathname, or a simple filename in the directory specified by the location parameter. (In the latter case the filename will be case sensitive.)

A simple filename may be a maximum of 12 characters, and a full pathname 255 characters.

- *location* is a numeric value specifying the directory where the file is located IF *file* is a simple filename:

Value	Directory
0	Central *SWDIR***lists** directory
1	User's *SWDIR***queues***username* directory

- *limit* is a numeric value specifying the maximum number of validations to add to the validations list (subject to the overall limit specified by the MAXVLD entry in the *SWDIR***etc****staffcfg** file, or 1000 if there is no such entry).
- *format* is a numeric value specifying the format of the file. *This is not currently implemented.*

Value	File format
0	UNIX

Value	File format
1	DOS

This affects line terminator characters.

- *charset* is a numeric value specifying the character set of *file*:

Value	Character set
0	SICS (Staffware Internal Character Set)
1	EUC (UNIX) Kanji
2	Shift-JIS (MS-Windows) Kanji
3	Unicode

Returns The number of validations added to the validations list - not normally used.

The list is created when the form is first displayed, and cannot be regenerated while the form is open.

VLDQUERY

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Passes a query to an integrated database (for example, Oracle) and adds the resulting values to the validations list for the current field.

There is no action if the iProcess Engine is not integrated with a database.

Syntax

```
vldquery (query, limit)
```

where:

- *query* is a text string specifying an SQL query.
- *limit* is a numeric value specifying the maximum number of validations to add to the validations list (subject to the overall limit specified by the MAXVLD entry in the *SWDIR***etc****staffcfg** file, or 1000 if there is no such entry).

Returns The number of validations added to the validations list - not normally used.

Examples

TIBCO iProcess Modeler:

```
vldquery ("select partno from parts where type=3", 50)
```

TIBCO Business Studio:

```
IPEValidationUtil.VLDQUERY("select partno from parts where
type=3", 50);
```

Chapter 11 Dynamic Data Exchange (DDE) Functions

Dynamic Data Exchange (DDE) is a method of transferring data between two Windows applications while they are running.

A DDE conversation is initiated by the DDE client, in this case the iProcess Workspace (Windows), by sending a message to the DDE server, for instance an application from which data is being requested, or which is being instructed to perform some task like opening an image window.

DDE facilities are mainly used in scripts, since multiple DDE statements are usually necessary for each interaction.

The Initiate Conversation message is sent with the DDEINITIATE script statement. This is followed by one or more messages such as DDEREQUEST, DDEEXECUTE or DDEGETTOPIC. The conversation is terminated with DDETERMINATE (or DDETERMALL).

For example:

```
   WINRUN ("C:\Program Files\Microsoft Office\Office10\EXCEL.exe
C:\BUDG.XLS", 1)

   init := DDEINITIATE (excelch, "EXCEL", "c:\BUDG.XLS")
   req  := DDEREQUEST (excelch, "R4C6", field1, 1, 5)
   exec := DDEPOKE (excelch, "R1C2", "Hello", 5)
   term := DDETERMINATE (excelch)
```

where *excelch* and *field1* are iProcess fieldnames.

In this example, iProcess Workspace (Windows) establishes a DDE conversation with Excel, instructs Excel to open the **BUDG.XLS** spreadsheet, and then retrieves the value of a particular cell into an iProcess field.

All DDE commands are performed synchronously - iProcess waits until the command completes before continuing.

The following DDE functions can be used.

Function	Usage	Description	See
DDEEXECUTE		Send an EXECUTE command to a server	page 143
DDEGETNAME		Get a DDE server's application name	page 144

Function	Usage	Description	See
DDEGETTOPIC		Get a DDE server's topic name	page 145
DDEINITIATE		Initiate a conversation with a DDE server	page 146
DDEPOKE		Send data to a server	page 148
DDEREQUEST		Request an item of data from a server	page 149
DDETERMALL		Terminate all DDE conversations	page 151
DDETERMINATE		Terminate a DDE conversation	page 152

DDEEXECUTE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Sends an EXECUTE command to the server on the specified channel.

Syntax

```
ddeexecute (channel, command, timeout)
```

where:

- *channel* is a numeric value specifying the channel number of the conversation.
- *command* is a text string specifying the command to EXECUTE.
- *timeout* is a numeric value specifying the number of seconds to wait for a response.

Returns One of the following numeric values:

Value	Description
0	Success
1	Not processed (bad command, or server does not handle EXECUTE)
2	Timeout - server busy
3	Bad channel
4	Unknown error from server

Examples

TIBCO iProcess Modeler:

```
ddeexecute (mychan, "RUN", 5)
```

TIBCO Business Studio:

```
IPEDDEUtil.DDEEXECUTE(mychan, "RUN", 5);
```

In this example, the channel number is specified as the integer data field **mychan**.

DDEGETNAME

Usage

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Get the DDE server application's name. (This function would be used if no server had been specified to DDEEXECUTE.)

Syntax

```
ddegetname (channel, fldret)
```

where:

- *channel* is a numeric value specifying the channel number of the conversation.
- *fldret* is a text string specifying the iProcess fieldname reference to contain the returned name.

Returns

One of the following numeric values:

Value	Description
0	Success
1	Bad channel

Examples

TIBCO iProcess Modeler:

```
ddegetname (mychan, myfield)
```

TIBCO Business Studio:

```
IPEDDEUtil.DDEGETNAME(mychan, myfield);
```

The arguments are passed as integer and string data fields respectively.

DDEGETTOPIC

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Get the DDE conversation's topic name. (This function would be used if no topic had been specified to DDEEXECUTE.)

Syntax

```
ddegettopic (channel, fldret)
```

where:

- *channel* is a numeric value specifying the channel number of the conversation.
- *fldret* is a text string specifying the iProcess fieldname reference to contain the returned name.

Returns One of the following numeric values:

Value	Description
0	Success
1	Bad channel

Examples

TIBCO iProcess Modeler:

```
ddegettopic (mychan, myfield)
```

TIBCO Business Studio:

```
IPEDDEUtil.DDEGETTOPIC(mychan, myfield);
```

The arguments are passed as integer and string data fields respectively.

DDEINITIATE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Initiates a DDE conversation with a DDE server.

Syntax

```
ddeinitiate (fldresult, server, topic)
```

where:

- *fldresult* specifies the name of the iProcess field to contain the resulting channel number if successful; this must be a numeric field of at least 10 digits (no decimals required).
- *server* is a text string representing the DDE server to talk to.
- *topic* is a server-specific text string representing the topic for the conversation.

The *topic* or the *server* and *topic* may be null strings (""). In the case of a null topic, the first available topic specified by the server is used, and this may be determined with the DDEGETTOPIC command. In the case of a null *server* (as well as a null *topic*), the first available server responds, and its name may be determined with the DDEGETNAME command.

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure (no such server/topic
2	Failure (result field too small)
3	Failure (not iProcess Workspace (Windows))

Examples

TIBCO iProcess Modeler:

```
ddeinitiate(excelch, "EXCEL", "FILE.XLS")
ddeinitiate(excelch, "WINWORD", FILEFLD + ".DOC")
```

TIBCO Business Studio:

```
IPEDDEUtil.DDEINITIATE(excelch, "EXCEL", "FILE.XLS");
IPEDDEUtil.DDEINITIATE(excelch, "WINWORD", FILEFLD + ".DOC");
```

DDEPOKE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Send some data to a DDE server.

Syntax

```
ddepoke (channel, item, data, timeout)
```

where:

- *channel* is a numeric value specifying the channel number of the conversation.
- *item* is a server-specific text string representing the data item being sent.
- *data* is a server-specific text string which is the data being sent.
- *timeout* is a numeric value specifying the number of seconds to wait for a response.

Returns One of the following numeric values:

Value	Description
0	Success
1	Not processed (bad item, or server does not handle POKE)
2	Timeout - server busy
3	Bad channel
4	Unknown error from server

Examples

TIBCO iProcess Modeler:

```
ddepoke (mychan, "COMM.BMK", strfield, 5)
```

TIBCO Business Studio:

```
IPEDDEUtil.DDEPOKE (mychan, "COMM.BMK", strfield, 5);
```

DDEREQUEST

Usage TIBCO iProcess Workspace (Windows)

Send a request to a DDE server for an item of data.

Syntax

```
dderequest (channel, item, flditem, format, timeout)
```

where:

- *channel* is a numeric value specifying the channel number of the conversation.
- *item* is a server-specific text string representing the data item requested.
- *flditem* is a text string specifying the iProcess fieldname reference to contain the returned string.
- *format* is one of the following numeric values, specifying the formatting requirement for the returned string:

Value	String formatting
0	All characters are to be placed into the iProcess field (flditem).
1	Truncate the string at first non-printing or non-ASCII character.

- *timeout* is a numeric value specifying the number of seconds to wait for a response.

Returns One of the following numeric values:

Value	Description
0	Success
1	Not processed (bad item or server does not handle REQUEST)
2	Timeout - server busy
3	Bad channel
4	Unknown error from server

Examples

TIBCO iProcess Modeler:

```
dderequest (mychan, "DATA5", strfield, 0, 5)
```

TIBCO Business Studio:

```
IPEDDEUtil.DDEREQUEST(mychan, "DATA5", strfield, 0, 5);
```

DDETERMALL

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Terminate all DDE conversations for this Form Window.

Syntax

```
ddetermall ()
```

Returns One of the following numeric values:

Value	Description
0	Success
1	Failure

Examples

TIBCO iProcess Modeler:

```
ddetermall ()
```

TIBCO Business Studio:

```
IPEDDEUtil.DDETERMALL();
```

DDETERMINATE

Usage TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Terminates a DDE conversation.

Syntax

```
ddeterminate (channel)
```

where *channel* is the channel number of the conversation to terminate.

Returns One of the following numeric values:

Value	Description
0	Success
> 0	Failure

Examples

TIBCO iProcess Modeler:

```
ddeterminate (excelch)
```

TIBCO Business Studio:

```
IPEDDEUtil.DDETERMINATE(excelch);
```

Chapter 12 Calling Scripts

The following functions can be used to call an iProcess script.

iProcess script objects are not supported in TIBCO Business Studio; therefore neither of the functions described in this chapter have equivalents in TIBCO Business Studio.

Function	Usage	Description	See
CALL		Run an iProcess script	page 154
SCRIPT		Run an iProcess script that can have a number of arguments defined and that may return a value	page 155

CALL

Usage

 TIBCO iProcess Workspace (Browser)

 TIBCO iProcess Workspace (Windows)

This can be used to run a script that is available to the procedure. See "Using Scripts" in the *TIBCO iProcess Modeler - Advanced Design* guide for more information about scripts.

This function can also be used within a script to call another script. This enables you to call a script, run it and then return back to the original script. You can also recursively call other scripts up to the maximum limit defined by the MAX_SCRIPT_CALL_DEPTH parameter in *SWDIR***etc****staffcfg**. For example, when defining a script called script1, you can use the CALL function to call script2. script2 can call script3 and script3 can call script4 and so on.

Syntax

```
call (scriptname)
```

where *script* is a text string specifying the name of the script being called.

Returns

One of the following numeric values:

Value	Description
-1	Error executing or error with syntax when checking the script
0	Script not found
1	Success

Example

```
call ("myscript")
```

SCRIPT

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

This can be used to run a script that is available to the procedure. See "Using Scripts" in the *TIBCO iProcess Modeler - Advanced Design* guide for more information about scripts.

This function uses iProcess variables to enable parameters to be input into a script ($ARG*n*) and an application defined returned value to be output from a script ($RETURN), when the script is executed. Returning a value from a script is useful if you want to map values to sub-procedure input parameters when a script is executed. See "Defining a Sub-Procedure" in the *TIBCO iProcess Modeler - Advanced Design* guide for more information about sub-procedures.

The only way to pass parameter values into a script is to use the SCRIPT function.

Scripts use the following iProcess variables:

- $ARG*n* (text)

  The format of the iProcess variable is $ARG*n* where *n* is a positive integer. If parameters have been specified for the script, the value of the fields for that parameter are represented as a string. This means that the original values will have to be converted back to their original types within the script. See Conversion Functions on page 43 for information on iProcess functions that can be used to convert data to different formats.

  If insufficient arguments have been supplied, the value of the input parameter is SW_NA.

- $RETURN

  $RETURN is treated as an iProcess field of variable type. It can have a value assigned to it or $RETURN can be used with an expression. When the script has finished executing, the value of this variable is used as the return value of the script. If a value has not been assigned during the execution of the script, the return value is the value of the last expression executed within the script.

Syntax

```
SCRIPT (scriptname [, param1, ...])
```

where:

- *scriptname* is a text string specifying the name of the script being called.

This must be in quotation marks. For example, if the scriptname is called AUTOBAL, then it must be entered in the expression as "AUTOBAL".

- *param* is used to define one or more input parameters to the script. These are converted to text type and referenced using $ARG*n* variable names in the script (where $ARG1 is the first parameter, $ARG2 is the second parameter and so on).

Returns One of the following numeric values:

Value	Description
0	Specified script not found
-1	Error executing or error with syntax when checking the script
Anything Else	The resulting data from the $RETURN (vartype) variable.

Example For a procedure with the following script called **ADDTOBAL** that takes 3 arguments (customer name, account balance and an amount to credit the balance by):

```
; Add credit ($ARG2) to balance ($ARG3)
NEWBAL := NUM($ARG2) + NUM($ARG3)
; Create the return string
; customer name($ARG1) : NEWBAL)
$RETURN := $ARG1 + ": " + STR(NEWBAL)
; END OF SCRIPT
;
```

The script returns a single string in the format **Customer Name: New Balance**.

The following expression:

```
CUSBAL := SCRIPT ("ADDTOBAL", CUSNAME, BALANCE, CREDIT)
```

will call the script and when the script has executed, the CUSBAL field will have a value in the format **CUSTOMER NAME:** ***new balance***. For example, **John Smith: 325**.

Chapter 13 Database Functions

The following function can be used to write fields within a work item to a table in the iProcess database on the server.

Function	Usage	Description	See
DBWRITEFIELDS		Write specified fields within a work item to a table in the iProcess database on the server.	page 158

DBWRITEFIELDS

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Write specified fields within a work item to a table in the iProcess database on the server.

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Syntax

```
DBWriteFields (TableName, IDString, FieldList, ExcludeList,
Flags)
```

where:

- *TableName* is a text string giving the name of a table in the iProcess database.
- *IDString* is user defined text that can be used to identify the records created in the database by this invocation of DBWRITEFIELDS.
- *FieldList* is a text string specifying the list of fields to be written, separated by commas. Wildcard characters * and ? can be included.
- *ExcludeList* is a text string specifying the list of fields NOT to be written, even though selected in *FieldList*. Wildcard characters * and ? can be included.
- *Flags* is one or more of the following numeric values:

Value	Behavior
0	Default behavior
1	Write fields marked as changed only
2	Write fields with data only
4	Remove existing records first
8	Ignore step name

Returns One of the following numeric values:

Value	Description
> 0	The number of field value records written to the table **Note: Memo and attachment fields are not written to the database.**
0	No fields matched include list and/or flags
-1	Function not supported by server
-2	Unspecified system error
-3	Failed to allocate FILDBWF session
-4	Function not supported in this context

Table Layout The table that is written to by the DBWRITEFIELDS expression must have the following layout:

- On a **Windows SQL Server** database:

```
TABLE swpro.DbFieldData(
node_id             INTEGER              NOT NULL,
proc_id             INTEGER              NOT NULL,
casenum             INTEGER              NOT NULL,
stepname            VARCHAR(8)           NULL,
id_string           VARCHAR(255)         NULL,
field_name          VARCHAR(31)          NOT NULL,
field_value         VARCHAR(255)         NULL,
field_flags         INTEGER              NOT NULL)
```

- On an **Oracle** database:

```
TABLE swpro.DbFieldData(
node_id             NUMBER(5)            NOT NULL,
proc_id             NUMBER(5)            NOT NULL,
casenum             NUMBER(10)           NOT NULL,
stepname            VARCHAR2(8)          NULL,
id_string           VARCHAR2(255)        NULL,
field_name          VARCHAR2(31)         NOT NULL,
field_value         VARCHAR2(255)        NULL,
field_flags         NUMBER(10)           NOT NULL)
```

The table needs to have an index constructed from **node_id**, **proc_id**, **casenum** and **field_name**.

The table must be created/owned by the iProcess database background user and the iProcess database foreground user must have select, insert, update and delete permissions.

Examples

TIBCO iProcess Modeler:

For a procedure with the following set of fields:

```
Currbalance, Retcode
Item01, Item02, Item03, Item04, Item05
Aaval1, Aaval2, Abval1, Abval2, Acval1, Acval2, Adval1, Adval2
```

The following expression:

```
Retcode := DbWriteFields ("CaseDataSnapshots", SW_USER:NAME,
"currbalance,itemval*,a?val*", "acval*", 2)
```

will result in all fields that have data (i.e. are not SW_NA), except for fields **Adval1** and **Adval2**, being written to the table **swpro.CaseDataSnapshots** in the iProcess database. Each record will have associated the current step identifier and the user's name. Any existing records for this case, step and user, and fields **Adval1** or **Adval2** would not be modified or deleted.

To remove all the records added by a number of instances of the above expressions (perhaps for different users and steps within the procedure), the last step of the procedure could be an automatic step that executed the following expression:

```
Retcode := DbWriteFields ("CaseDataSnapshots", "", "", "", 12)
```

This would delete all records for this case, for any user name and for any step identifier.

TIBCO Business Studio:

The two database functions in the iProcess example can be replicated in TIBCO Business Studio as follows:

```
Retcode = IPEDatabaseUtil.DBWRITEFIELDS("CaseDataSnapshots",
IPEUserUtil.GETATTRIBUTE("NAME"), "currbalance,itemval*,a?val*",
"acval*", 2);
```

```
Retcode = IPEDatabaseUtil.DBWRITEFIELDS("CaseDataSnapshots", "",
"", "", 12);
```

Chapter 14 Procedure Functions

The following functions can be used to control the processing of cases.

Function	Usage	Description	See
CASESTART		Start a new case	page 162
TRIGGEREVENT		Trigger an event step1	page 164
CASECLOSE		Close a case	page 167
GOTOSTEP		Go to a step	page 168

CASESTART

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Start a new case of a procedure at a step, with input data.

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Syntax

```
casestart(procname, casedesc, startstep,
flags, [fieldname, fieldvalue] ...])
```

where:

- *procname* (text) is the procedure name; the default is the current procedure at the current node.
- *casedesc* (text) is the case description.
- *startstep* (text) is the step at which to start the case; if the start step is not specified then `""` must be supplied as the argument which defaults to the start step in the procedure definition.
- *flags* (numeric) is currently not used but must be specified. The argument value should be supplied as `0`.
- *fieldname* (text) is used to start the case with data. The argument consists of sets of pairs of arguments. Fieldname is the first argument in the pair.
- *fieldvalue* (anytype) is used to start the case with data. The argument consists of sets of pairs of arguments. Fieldvalue is the second argument in the pair.

This expression works with the default version of a procedure.

Returns

One of the following numeric values:

Value	Description
>0	Case Number
-100	Invalid or unknown procedure
-101	Invalid or unknown start step
-104	Start step is not a valid type

Value	Description
-105	Procedure requires a case description and none was supplied
-106	Specified procedure is a sub-procedure
-108	Unknown error from server

Examples

TIBCO iProcess Modeler:

1. This example starts a new case of the procedure **Hiring** with a case description of **Test Case** at the first step defined in the procedure definition.

```
casestart ("hiring","Test Case","",0)
```

2. This example starts a new case of the procedure **Hiring** with a case description which is a concatenation of text and the return value from the TIMESTR function, at a step called **ALTSTART.**

```
casestart ("hiring@node1", "Autostarted at" +
TIMESTR(SW_TIME), "ALTSTART",0)
```

3. This example starts a new case of the procedure **Hiring** with a case description of **Test Case** at the start step defined in the procedure definition with the following data:

```
casestart ("hiring", "Test Case","",0, "TEXTFLD", txtfld, \
"NUMFLD", numfld, \
"STARTDATE", SW_DATE, \
"STARTTIME", SW_TIME)
```

This example uses the "\" new line continuation feature available in iProcess scripts.

TIBCO Business Studio

1. Equivalent of iProcess Example 1:

```
IPEProcessUtil.CASESTART("hiring","Test Case","",0);
```

2. Equivalent of iProcess Example 2. Note the use of IPESystemValues for SW_TIME.

```
IPEProcessUtil.CASESTART("hiring@node1", "Autostarted at" +
IPEConversionUtil.TIMESTR(IPESystemValues.SW_TIME),
"ALTSTART",0);
```

3. Equivalent of iProcess Example 3:

```
IPEProcessUtil.CASESTART("hiring", "Test Case","",0, "TEXTFLD",
txtfld, "NUMFLD", numfld, "STARTDATE", IPESystemValues.SW_DATE,
"STARTTIME", IPESystemValues.SW_TIME);
```

TRIGGEREVENT

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Trigger an event step in a case of a procedure, with or without input data.

Syntax

```
triggerevent (procname, casenum, eventstep,
options, [fieldname, fieldvalue]...])
```

where:

- *procname* (text) is the procedure name.
- *casenum* (numeric) is the case number.
- *eventstep* (text) is the name of the event step to trigger.
- *options* (numeric) must be any combination of the values specified in the following table.

Value	Description
0	Do not use any of the following options.
1	Resurrect the (previously closed) case.
2	Update the **pack_data** table for the case using the supplied *fieldname/fieldvalue* pairs. You can propagate these **pack_data** changes into sub-cases as described in the section “Propagation of New Field Values” in the *TIBCO iProcess Modeler Integration Techniques* guide.
4	Recalculate deadlines for the case, *but not for* any associated sub-cases, using the supplied *fieldname/fieldvalue* pairs. See "Dynamically Recalculating Deadlines" in the *TIBCO iProcess Modeler - Basic Design* guide for more information about using this option.

Value	Description
8	Recalculate deadlines for the case, *and for* any associated sub-cases, using the supplied *fieldname*/*fieldvalue* pairs. See "Dynamically Recalculating Deadlines" in the *TIBCO iProcess Modeler - Basic Design* guide for more information about using this option.

For example, if you want to recalculate deadlines for the case and its sub-cases (8) and update pack data (2), specify the *options* value as **10**. If you do not want to resurrect a closed case, update pack data or recalculate deadlines, specify the *options* value as **0**.

- *fieldname* (text) is the name of a field that you want to update with the value specified in the following *fieldvalue* parameter.
- *fieldvalue* (anytype) is the new value that you want to specify for the field specified in the preceding *fieldname* parameter.

You can supply as many *fieldname*/*fieldvalue* pairs as you require. You must supply at least one *fieldname*/*fieldvalue* pair if you have specified an *options* value to update pack data or recalculate deadlines for the case.

Returns

One of the following numeric values:

Value	Description
1	Success
-100	Invalid or unknown procedure
-101	Invalid or unknown event step
-102	Invalid or unknown case number
-104	Event is not a valid type
-107	Specified case has terminated
-108	Unknown error from server
-109	Invalid or unknown parameter

Examples

TIBCO iProcess Modeler:

1. This example triggers the step called **Event** of the procedure **Hiring** in case **23**.

```
triggerevent ("hiring", 23, "event",0)
```

2. This example triggers the step called **Event** in the procedure **Hiring** in case **23 and** inputs the values **John** in the FIRST (name) field and **Smith** in the LAST (name) field.

```
triggerevent ("hiring", 23, "event",0,"FIRST","John",
"LAST","Smith")
```

TIBCO Business Studio

1. Equivalent of iProcess Example 1:

```
IPEProcessUtil.TRIGGEREVENT("hiring", 23, "event",0);
```

2. Equivalent of iProcess Example 2:

```
IPEProcessUtil.TRIGGEREVENT("hiring", 23,
"event",0,"FIRST","John", "LAST","Smith");
```

CASECLOSE

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine. It will return SW_NA.

Close a case of a procedure.

Syntax

```
caseclose (procname, casenum)
```

where:

- *procname* (text) is the procedure name.
- *casenum* (numeric) is the case number.

Returns

One of the following numeric values:

Value	Description
1	Success
-100	Invalid or unknown procedure
-101	Invalid or unknown case number
-107	Specified case has terminated
-108	Unknown error from server

Examples

TIBCO iProcess Modeler:

This example closes case **23** of the procedure **Hiring**.

```
caseclose ("hiring", 23)
```

TIBCO Business Studio

```
IPEProcessUtil.CASECLOSE("hiring", 23);
```

GOTOSTEP

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

(iProcess only) This expression is not available to the TIBCO iProcess Script Server Plug-in. Therefore, even though you can successfully enter the expression in your iProcess Script plug-in definition, it will not be processed by the iProcess Engine and returns a -110 error.

On release of the current step, you can jump to another step. You can either process the current step or not.

Syntax

```
gotostep (step, extra, flags)
```

where:

- *step* (text) is the step name to go to.
- *extra* (text) is currently ignored but may be used to modify the behavior of this function in the future.
- *flags* (numeric) is the following numeric value:

Value	Behavior
0	The specified step is processed and the current step's actions are processed.
1	The specified step is processed but the current step's actions are NOT processed.

The GOTOSTEP expression is only valid in the context of an open work item. If the work item is not released at the end of the current session, the GOTOSTEP will be ignored. This means that if the expression is executed and the work item is kept, and then re-opened and released, the GOTOSTEP will have no effect.

Returns

One of the following numeric values:

Value	Description
1	Success

Value	Description
-101	Invalid or unknown step name
-102	Invalid or unknown case number
-104	Step is not a valid type
-108	Unknown error from server
-110	Function not supported in this context

Examples

TIBCO iProcess Modeler:

1. This example processes the step called **Gotostep** on release of the current step in the procedure. The actions of the current step are not actioned:

   ```
   gotostep ("gotostep","",1)
   ```

2. This example processes the step called **Gotostep** on release of the current step in the procedure. The actions of the current step are actioned:

   ```
   gotostep ("gotostep","",0)
   ```

TIBCO Business Studio

1. Equivalent of iProcess Example 1:

   ```
   IPEProcessUtil.GOTOSTEP("gotostep", "", 1);
   ```

2. Equivalent of iProcess Example 2:

   ```
   IPEProcessUtil.GOTOSTEP("gotostep", "", 0);
   ```

Chapter 15 Array Functions

The following functions are used to identify elements within array fields.

Function	Usage	Description	See
FINDARRELEMENT		Returns the index number of the next array element that matches the given value in the given array field after the given start element.	page 172
NEXTARRELEMENT		Returns the index number of the next assigned array element in the given array field after the given start element.	page 174

FINDARRELEMENT

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Returns the index number of the next array element that matches the given value in the given array field after the given start element. See "Using Arrays Fields" in the *TIBCO iProcess Modeler - Advanced Design* guide for more information about using array fields.

Syntax

```
findarrelement (arrayname, startelement, value)
```

where:

- *arrayname* (text) is the name of the array field on which to perform the search.

This must be in quotation marks. For example, if the array field name is CUSTOMER, then it must be entered in the expression as "CUSTOMER". For composite fields, only the composite field name should be entered (without any sub-field definition).

- *startelement* (numeric) element index number to start the search from or **-1** for the first assigned element.
- *value* (text, numeric, date or time) is the value to find in the array elements.

Returns

One of the following numeric values:

Value	Description
-1	No more elements with the given value are found after start element
Anything Else	Numeric value for the index number of the array element

Examples

TIBCO iProcess Modeler:

For a procedure with the following array field:

```
custname[0],J Smith
custname[1],J Brown
custname[3],T Jones
```

and the following numeric field:

```
custidx
```

The following expression:

```
custidx := findarrelement (“CUSTOMER”, -1, “J Brown”)
```

sets the field `custidx` to the index of the element that contains the value **J Brown** in the `custname` array. It starts the search at the first element in the array and searches the array until it finds the value. `custidx` will be set to a value of 1.

TIBCO Business Studio

```
custidx = IPEArrayUtil.FINDARRELEMENT("custname", -1, "J Brown");
```

NEXTARRELEMENT

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Returns the index number of the next assigned array element in the given array field after the start element. This is useful when an application does not store data in contiguous array elements. See "Using Arrays Fields" in the *TIBCO iProcess Modeler - Advanced Design* guide for more information about using array fields.

Syntax

```
nextarrelement(arrayname, startelement)
```

where:

- *arrayname* (text) is the name of the array field on which to perform the search.

This must be in quotation marks. For example, if the array field name is CUSTOMER, then it must be entered in the expression as "CUSTOMER". For composite fields, only the composite field name should be entered (without any sub-field definition).

- *startelement* (numeric) is the element index number to start the search from or **-1** for the first assigned element.

Returns

One of the following numeric values:

Value	Description
-1	No more array elements are found after start element
Anything Else	Numeric value for the index number of the array element

Examples

TIBCO iProcess Modeler:

For a procedure with the following array field:

```
custname[0],J Smith
custname[1],J Brown
custname[3],T Jones
```

and the following numeric field:

```
custidx
```

and the following text field:

```
custname
```

The following script will build a comma separated list of all the customer names:

```
custidx := -1
custlist := ""
while ((custidx:= nextarrelement ("custname", custidx))<>-1)
if (strlen (custlist) >0)
custlist := custlist + ","
endif
custlist:= custlist + custname [custidx]
wend
```

TIBCO Business Studio

```
while ((custidx = IPEArrayUtil.NEXTARRELEMENT("custname",
      custidx))!=-1)
{  if (IPEStringUtil.STRLEN(custlist) >0)
      {custlist = custlist + ",";
      }
       else
         {custlist = custlist + custname [custidx];
         }
}
```

Chapter 16 General Utility Functions

The following functions are general utility functions.

Function	Usage	Description	See
SELECTVAL		Evaluate an expression that returns a boolean type result	page 178
SWITCHVAL		Evaluate an expression that only returns a numeric type result that corresponds to an argument number	page 180

SELECTVAL

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Evaluates a conditional argument that returns data as a result of the Boolean type result (i.e. true or false).

Syntax

```
selectval (evalexpression, trueexpression,
falseexpression)
```

where:

- *evalexpression* (Boolean) is an expression to evaluate that returns a Boolean value.
- *trueexpression* (vartype) is the expression result to return if the result of the evaluating *evalexpression* is true.
- *falseexpression* (vartype) is the expression result to return if the result of evaluating *evalexpression* is false.

Both trueexpression and falseexpression are evaluated so they should not contain any side effects. For example, in the following expression:

Both *trueexpression* and *falseexpression* are evaluated so they should not contain any side effects. For example, in the following expression:

```
num1:=selectval(a>b,casestart(proca),
casestart(procb))
```

the value given to `num1` is the result of one of the casestart functions (depending on the condition) but both the casestart functions are executed i.e. new cases of both proca and procb are started. This means you should avoid using functions like casestart where an action is performed as a result of the function.

Returns

The result data from the evaluation is either *trueexpression* or *falseexpression*. The return from the function is a Vartype.

SELECTVAL and SWITCHVAL return Vartype. This means that SELECTVAL and SWITCHVAL can only be used as part of an assignment or in sub-procedure call definition and call expressions.

Examples

TIBCO iProcess Modeler:

Assign the contents of **Field2** with the result of the SELECTVAL expression on **Field1**.

```
field2:= selectval (field1 = "New Patient",1,0)
```

If **Field1** does equal **New Patient** then **1** is assigned to **Field2**. If **Field1** does not equal **New Patient** then **0** is assigned to **Field2**.

TIBCO Business Studio

```
field2 = IPEGeneralUtil.SELECTVAL(field1 = "New Patient",1,0);
```

SWITCHVAL

Usage

TIBCO iProcess Workspace (Browser)

TIBCO iProcess Workspace (Windows)

Evaluates a conditional argument that returns a numeric type result based on a range of arguments.

Syntax

```
switchval (numexpression, defaultvalue,
case1val, case2val [case3val, case4val ...])
```

where:

- *numexpression* (real) is an expression to evaluate that returns a positive numeric integer value of 1 to *n*.
- *defaultvalue* (vartype) is the result to return if the result of the evaluating *numexpression* is neither a positive integer nor a value between 1 to *n*, where *n* is the number of case arguments provided in the expression.
- *case1val* (vartype) is a case argument whose result is returned if the *numexpression* is a value between 1 to n, where *n* is the number of case arguments provided in the expression.
- *case2val* (vartype) is a case argument whose result is returned if the *numexpression* is a value between 2 to n, where *n* is the number of case arguments provided in the expression.
- *case3val* (vartype) is a case argument whose result is returned if the *numexpression* is a value between 3 to n, where *n* is the number of case arguments provided in the expression.
- *case4val* (vartype) is a case argument whose result is returned if the *numexpression* is a value between 4 to n, where *n* is the number of case arguments provided in the expression.

At least four arguments must be provided. Additional arguments are optional.

Returns

The resulting data from the evaluation of the *numexpression* and *defaultexpression*.

Examples

TIBCO iProcess Modeler:

This example converts a number to a name from a list of names:

```
strfield := switchval (numfield, "Out Of
Range", "John", "Richard", "Michael", "Mark",
"Steven", "Paul")
```

If the value of numfield is 1, the function returns the value **John**. If the value of numfield is 4, the function returns the value **Mark**. If the value is < 1 or > 6 then the function returns **Out Of Range**.

TIBCO Business Studio

```
strfield = IPEGeneralUtil.SWITCHVAL (numfield, "Out Of
Range", "John", "Richard", "Michael", "Mark", "Steven", "Paul");
```

Chapter 17 TIBCO Business Studio JavaScript Classes

The following JavaScript classes are used within TIBCO Business Studio.

Function	Usage	Description	See
IPEProcessNameUtil		When passed a TIBCO Business Studio Process name, converts it to an iProcess procedure name.	page 184
IPETaskNameUtil		When passed a TIBCO Business Studio Task name, converts it to an iProcess step name.	page 185
IPEGroupUtil		Allows you to access the group queue from which the work item was chosen (User Task only).	page 186
IPEStarterUtil		Allows you to access the starter of a case.	page 187
IPEUserUtil		Allows you to access the current user (User Task only).	page 188

IPEProcessNameUtil

Usage

 TIBCO Business Studio

Converts valid TIBCO Business Studio Process names into equivalent iProcess procedure names. For example, long Process names are truncated as they would be upon deployment to iProcess. This ensures that the Sub-Process Call Task works as expected upon deployment or export. This class should be used to populate the string array field that is used to create a dynamic sub-process in TIBCO Business Studio (see *TIBCO Business Studio iProcess Developer's Guide*).

The method takes a string literal.

Syntax

```
IPEProcessNameUtil.GETPROCESSNAME("studioprocname");
```

where *studioprocname* (string literal) is the name of the TIBCO Business Studio process name that you want to convert to an iProcess procedure name. TIBCO Business Studio cannot validate whether the value you pass equates to a valid process name in the runtime environment.

iProcess Translation

A string that contains the expected iProcess procedure name upon export or deployment.

Example

IPEProcessNameUtil.GETPROCESSNAME("TESTPROCEDURE1");

Converted to TESTPROC when executed in iProcess.

IPETaskNameUtil

Usage

 TIBCO Business Studio

Converts valid TIBCO Business Studio Task names into equivalent iProcess step names. For example, long Task names are truncated. This is useful for example if you need to populate a start step name array when setting up a dynamic sub-process in TIBCO Business Studio (see *TIBCO Business Studio iProcess Developer's Guide*).

Syntax

```
IPEProcessNameUtil.GETTASKNAME("procname","taskname");
```

where:

- *procname* (string literal) is the name of the TIBCO Business Studio process that contains the task name that you want to convert to an iProcess step name.
- *taskname* (string literal) is the name of the TIBCO Business Studio task name that you want to convert to an iProcess step name.

TIBCO Business Studio cannot validate whether the values you pass equate to valid process or task names in the runtime environment.

iProcess Translation

A string that contains the expected iProcess step name upon export or deployment.

Example

```
IPEProcessNameUtil.GETTASKNAME("PROC1","COLLECTDETAILS");
```

Converted to COLLECTD when executed in iProcess.

IPEGroupUtil

Usage

 TIBCO Business Studio

Allows you to access the group queue from which the work item was chosen from within a TIBCO Business Studio script (User Task scripts only).

Syntax

```
IPEGroupUtil.GETATTRIBUTE("NAME");
```

where *NAME* (string literal) is the attribute that you want to access.

Example

MyField=IPEGroupUtil.GETATTRIBUTE("NAME");

This is converted to iProcess syntax (MyField := SW_GROUP:NAME) in iProcess and assigns MyField to the name of the group queue from which the work item was chosen.

IPEStarterUtil

Usage

 TIBCO Business Studio

Allows you to access the starter of a case from within a TIBCO Business Studio script.

Syntax

```
IPEStarterUtil.GETATTRIBUTE("NAME");
```

where *NAME* (string literal) is the attribute that you want to access.

Example

MyField=IPEStarterUtil.GETATTRIBUTE("NAME");

This is converted to iProcess syntax (MyField := SW_STARTER:NAME) in iProcess and assigns MyField to the starter of the case.

IPEUserUtil

Usage

TIBCO Business Studio

Allows you to access the current user from within a TIBCO Business Studio script (User Task scripts only).

Syntax

```
IPEUserUtil.GETATTRIBUTE("NAME");
```

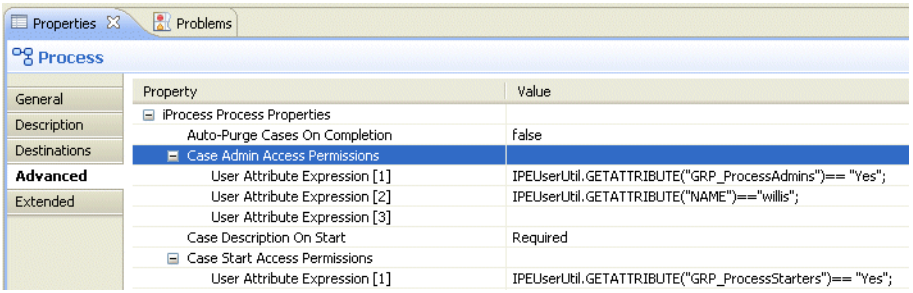
where *NAME* (string literal) is the attribute that you want to access.

Examples

MyField=IPEUserUtil.GETATTRIBUTE("NAME");

This is converted to iProcess syntax (MyField := SW_USER:NAME) in iProcess and assigns MyField to the current user.

Another use of this class is to specify case admin or case start access permissions on the Advanced tab in the Properties view for a Process:



This example limits case administration to only users in the **ProcessAdmins** group and the user **willis**. Case starts can only be performed by users in the **ProcessStarters** group.

Index

Symbols

A

B

C

D

E

F

G

H

I

M

N

O

P

R

S

T

U

V

W

Y