



- ▶ Version 10.1
- ▶ Synchronising Threads
- ▶ Get Tokens, Put Tokens, Token Pool and Token Requests

Dyalog Ltd
 South Barn
 Minchens Court
 Minchens Lane
 Bramley
 Hampshire
 RG26 5BH
 United Kingdom

Phone:
 + 44 (0) 1256 830 030

Fax:
 + 44 (0) 1256 830 031

e-mail:
 sales@dyalog.com

Get Tokens

$\{R\} \leftarrow \{X\} \square TGET Y$

Y must be a simple integer scalar or vector that specifies one or more tokens, each with a specific non-zero token type, that are to be retrieved from the pool.

X is an optional time-out value in seconds.

Shy result R is a scalar or vector containing the values of the tokens of type Y that have been retrieved from the token pool.

Note that types of the tokens in the pool may be positive or negative, and the elements of Y may also be positive or negative.

A request ($\square TGET$) for a positive token will be satisfied by the presence of a token in the pool with the same positive or negative type. If the pool token has a positive type, it will be removed from the pool. If the pool token has a negative type, it will remain in the pool. Negatively typed tokens will therefore satisfy an infinite number of requests for their positive equivalents. Note that a request for a positive token will remove one if it is present, before resorting to its negative equivalent

A request for a negative token type will only be satisfied by the presence of a negative token type in the pool, and that token will be removed.

If, when a thread calls $\square TGET$, the token pool satisfies all of the tokens specified by Y , the function returns immediately with a (shy) result that contains the values associated with the pool tokens. Otherwise, the function will block (wait) until all of the requested tokens are present or until a timeout (as specified by X) occurs.

For example, if the pool contains only tokens of type 2:

$\square TGET 2 4 \square$ *blocks waiting for a 4-token ...*

The $\square TGET$ operation is atomic in the sense that no tokens are taken from the pool until all of the requested types are present. While this last example is waiting for a 4-token, other threads could take any of the remaining 2-tokens.

Note also, that repeated items in the right argument are distinct. The following will block until there are at least 3×2 -tokens in the pool:

$\square TGET 3/2 \square$ *wait for 3 × 2-tokens ...*



Version 10.1

Synchronising Threads

Get Tokens, Put Tokens, Token Pool and Token Requests

The pool is administered on a first-in-first-out basis. This is significant only if tokens of the same type are given distinct values.

For example:

```
□TGET □TPOOL a empty pool.
```

```
'ABCDE'□TPUT`2 2 3 2 3 a pool some tokens.
```

```
+□TGET 2 3
```

AC

```
+□TGET 2 3
```

BE

Timeout is signalled by the return of a scalar 0 token. By default, the value of a token is the same as its type. This means that, unless you have explicitly set the value of a token to 0, a `□TGET` result of 0 unambiguously identifies a timeout.

Beware - the following statement will wait forever and can only be terminated by a strong interrupt.

```
□TGET 0 a wait forever ...
```

Note too that if a thread waiting to `□TGET` tokens is `□TKILL`ed, the thread disappears without removing any tokens from the pool. Conversely, if a thread that has removed tokens from the pools is `□TKILL`ed, the tokens are not returned to the pool.

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com



- ▶ Version 10.1
- ▶ Synchronising Threads
- ▶ Get Tokens, Put Tokens, Token Pool and Token Requests

Put Tokens

$\{R\} \leftarrow \{X\} \quad \square TPUT \ Y$

Y must be a simple integer scalar or vector of non-zero token types.

X is an optional array of values to be stored in each of the tokens specified by Y .

Shy result R is a vector of thread numbers (if any) unblocked by the $\square TPUT$.

Examples

$\square TPUT \ 2 \ 3 \ 2 \ \# \text{ put a 2-token, a 3-token and another 2-token into the pool.}$

$88 \ \square TPUT \ 2 \ \# \text{ put another 2-token into the pool this token has the value 88.}$

$'Hello' \ \square TPUT \ ^4 \ \# \text{ put a ^4-token into the pool with the value 'Hello' .}$

If X is omitted, the value associated with each of the tokens added to the pool is the same as its type.

Note that you cannot put a 0-token into the pool; 0-s are removed from Y .

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com



- ▶ Version 10.1
- ▶ Synchronising Threads
- ▶ Get Tokens, Put Tokens, Token Pool and Token Requests

Token Pool

$R \leftarrow \square TPOOL$

R is a simple scalar or vector containing the token types for each of the tokens that are currently in the token pool.

The following ($\square ml = 0$) function returns a 2-column snapshot of the contents of the pool. It does this by removing and replacing all of the tokens, restoring the state of the pool exactly as before. Coding it as a single expression guarantees that snap is atomic and cannot disturb running threads.

```
snap ← { (□TGET w) { (□†w α) { α } α □TPUT w } w }
```

```
snap □TPOOL
```

```
1 hello world
2 2
3 2
2 three-type token
2 2
```

Token Requests

$R \leftarrow \square TREQ Y$

Y is a simple scalar or vector of thread numbers.

R is a vector containing the concatenated token requests for all the threads specified in Y . This is effectively the result of catenating all of the right arguments together for all threads in Y that are currently executing $\square TGET$.

Example

```
□TREQ □TNUMS n tokens required by all threads.
```

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com