1 APR::Status - Perl Interface to the APR_STATUS_IS_* macros
1.1 Synopsis

use APR::Status ();
eval { $obj->mp_method() };
if ($@ && $ref $@ eq 'APR::Error' && APR::Status::is_EAGAIN($@)) {
    # APR_STATUS_IS_EAGAIN(s) of apr_errno.h is satisfied
}

1.2 Description

An interface to apr_errno.h composite error codes.

As discussed in the APR::Error manpage, it is possible to handle APR/Apache/mod_perl exceptions in the following way:

eval { $obj->mp_method() };
if ($@ && $ref $@ eq 'APR::Error' && $@ == $some_code)
    warn "handled exception: $@";
}

However, in cases where $some_code is an APR::Const constant, there may be more than one condition satisfying the intent of this exception. For this purpose the APR C library provides in apr_errno.h a series of macros, APR_STATUS_IS_*\(^{1}\), which are the recommended way to check for such conditions. For example, the APR_STATUS_IS_EAGAIN macro is defined as

```c
#define APR_STATUS_IS_EAGAIN(s)         ((s) == APR_EAGAIN
|| (s) == APR_OS_START_SYSERR + ERROR_NO_DATA
|| (s) == APR_OS_START_SYSERR + SOCEWOULDBLOCK
|| (s) == APR_OS_START_SYSERR + ERROR_LOCK_VIOLATION)
```

The purpose of APR::Status is to provide functions corresponding to these macros.

1.3 Functions

1.3.1 is_EACCES

Check if the error is matching EACCES and its variants (corresponds to the APR_STATUS_IS_EACCES macro).

```perl
$status = APR::Status::is_EACCES($error_code);
```

- **arg1:** $error_code (integer or APR::Error object)

  The error code or to check, normally $@ blessed into APR::Error object.

- **ref:** $status (boolean)
- **since:** 2.0.00
An example of using \texttt{is\_EACCES} is when reading the contents of a file where access may be forbidden:

\begin{verbatim}
    eval { $obj->slurp_filename(0) }
    if ($@) {
        return Apache2::Const::FORBIDDEN
            if ref $@ eq 'APR::Error' && APR::Status::is_EACCES($@);
        die $@;
    }
\end{verbatim}

Due to possible variants in conditions matching \texttt{EACCES}, the use of this function is recommended for checking error codes against this value, rather than just using \texttt{APR::Const::EACCES} directly.

\subsection*{1.3.2 \texttt{is\_EAGAIN}}

Check if the error is matching \texttt{EAGAIN} and its variants (corresponds to the \texttt{APR\_STATUS\_IS\_EAGAIN} macro).

\begin{verbatim}
    $status = APR::Status::is_EAGAIN($error_code);
\end{verbatim}

- \textbf{arg1: $error\_code} (integer or \texttt{APR::Error} object)

  The error code or to check, normally $@ blessed into \texttt{APR::Error} object.

- \textbf{ret: $status} (boolean)
- \textbf{since: 2.0.00}

For example, here is how you may want to handle socket read exceptions and do retries:

\begin{verbatim}
    use APR::Status ();
    # ....
    my $tries = 0;
    my $buffer;
    RETRY: my $rlen = eval { $socket->recv($buffer, SIZE) }
    if ($@ && ref($@) && APR::Status::is_EAGAIN($@)) {
        if ($tries++ < 3) {
            goto RETRY;
        }
        else {
            # do something else
        }
    } else {
        # do something else
    }
    else {
        die "eval block has failed: $@";
    }
\end{verbatim}

Notice that just checking against \texttt{APR::Const::EAGAIN} may work on some Unices, but then it will certainly break on win32. Therefore make sure to use this macro and not \texttt{APR::Const::EAGAIN} unless you know what you are doing.
1.3.3 \texttt{is\_ENOENT}

Check if the error is matching ENOENT and its variants (corresponds to the APR\_STATUS\_IS\_ENOENT macro).

\begin{verbatim}
$status = APR::Status::is_ENOENT($error_code);
\end{verbatim}

- \textbf{arg1:} \texttt{$error\_code} (integer or APR::Error object)

  The error code or to check, normally @ blessed into APR::Error object.

- \textbf{ret:} \texttt{$status} (boolean)

- \textbf{since:} 2.0.00

An example of using \texttt{is\_ENOENT} is when reading the contents of a file which may not exist:

\begin{verbatim}
eval { $obj->slurp_filename(0) };
if ($@) {
   return Apache2::Const::NOT_FOUND
   if ref $@ eq 'APR::Error' && APR::Status::is_ENOENT($@);
   die $@;
}
\end{verbatim}

Due to possible variants in conditions matching ENOENT, the use of this function is recommended for checking error codes against this value, rather than just using APR::Const::ENOENT directly.

1.3.4 \texttt{is\_EOF}

Check if the error is matching EOF and its variants (corresponds to the APR\_STATUS\_IS\_EOF macro).

\begin{verbatim}
$status = APR::Status::is_EOF($error_code);
\end{verbatim}

- \textbf{arg1:} \texttt{$error\_code} (integer or APR::Error object)

  The error code or to check, normally @ blessed into APR::Error object.

- \textbf{ret:} \texttt{$status} (boolean)

- \textbf{since:} 2.0.00

Due to possible variants in conditions matching EOF, the use of this function is recommended for checking error codes against this value, rather than just using APR::Const::EOF directly.

1.3.5 \texttt{is\_ECONNABORTED}

Check if the error is matching ECONNABORTED and its variants (corresponds to the APR\_STATUS\_IS\_ECONNABORTED macro).
$status = APR::Status::is_ECONNABORTED($error_code);

- **arg1:** $error_code (integer or APR::Error object)
  - The error code or to check, normally @$_ blessed into APR::Error object.
- **ret:** $status (boolean)
- **since:** 2.0.00

Due to possible variants in conditions matching ECONNABORTED, the use of this function is recommended for checking error codes against this value, rather than just using APR::Const::ECONNABORTED directly.

### 1.3.6 is_ECONNRESET

Check if the error is matching ECONNRESET and its variants (corresponds to the APR_STATUS_IS_ECONNRESET macro).

$status = APR::Status::is_ECONNRESET($error_code);

- **arg1:** $error_code (integer or APR::Error object)
  - The error code or to check, normally @$_ blessed into APR::Error object.
- **ret:** $status (boolean)
- **since:** 2.0.00

Due to possible variants in conditions matching ECONNRESET, the use of this function is recommended for checking error codes against this value, rather than just using APR::Const::ECONNRESET directly.

### 1.3.7 is_TIMEUP

Check if the error is matching TIMEUP and its variants (corresponds to the APR_STATUS_IS_TIMEUP macro).

$status = APR::Status::is_TIMEUP($error_code);

- **arg1:** $error_code (integer or APR::Error object)
  - The error code or to check, normally @$_ blessed into APR::Error object.
- **ret:** $status (boolean)
- **since:** 2.0.00

Due to possible variants in conditions matching TIMEUP, the use of this function is recommended for checking error codes against this value, rather than just using APR::Const::TIMEUP directly.
1.4 See Also

mod_perl 2.0 documentation.

1.5 Copyright

mod_perl 2.0 and its core modules are copyrighted under The Apache Software License, Version 2.0.

1.6 Authors

The mod_perl development team and numerous contributors.
# Table of Contents:

1 | APR::Status - Perl Interface to the APR_STATUS_IS_* macros | 1 |
---|---|---|
1.1 | Synopsis | 2 |
1.2 | Description | 2 |
1.3 | Functions | 2 |
1.3.1 | is_EACCES | 2 |
1.3.2 | is_EAGAIN | 3 |
1.3.3 | is_ENOENT | 4 |
1.3.4 | is_EOF | 4 |
1.3.5 | is_ECONNABORTED | 4 |
1.3.6 | is_ECONNRESET | 5 |
1.3.7 | is_TIMEUP | 5 |
1.4 | See Also | 6 |
1.5 | Copyright | 6 |
1.6 | Authors | 6 |