### 22.3.6 The Catch operation

The **catch** operation is used to catch exceptions.

***Syntactical Structure***

( *Port* | **any** **port** | **any from** PortArrayRef ) "." **catch**

[ "(" ( *Signature* "," *TemplateInstance* ) | *TimeoutKeyword* ")" ]

[ **from** *Address* ]

[ "->" [ **value** *VariableRef* ]

[ **sender** *VariableRef* ]

[ @**index** **value** *VariableRef* ] ]

NOTE: *Address* may be an *AddressRef*, a list of *AddressRef*-s or "**any component**".

***Semantic Description***

The **catch** operation is used to catch exceptions raised by a test component or the SUT as a reaction to a procedure call. Exceptions are specified as types and thus, can be treated like messages, e.g. templates can be used to distinguish between different values of the same exception type.

The **catch** operation removes the top exception from the associated incoming port queue if, and only if, that top exception satisfies all the matching criteria associated with the **catch** operation.

A **catch** operation may be restricted to a certain communication partner in case of one-to-many connections. This restriction shall be denoted by using the **from** keyword.

The (optional) redirection part of the **catch** operation comprises of storing the exception value and the retrieval of the address of the calling component. The keyword **value** is used to retrieve the value of an exception and/or the parts of it and the keyword **sender** is used when it is required to retrieve the address of the sender.

The **catch** operation may be part of the response and exception handling part of a **call** operation or be used to determine an alternative in an **alt** statement. If the **catch** operation is used in the accepting part of a **call** operation, the information about port name and signature reference to indicate the procedure that raised the exception is redundant, because this information follows from the **call** operation. However, for readability reasons (e.g. in case of complex **call** statements) this information shall be repeated.

**The Timeout exception**

There is one special **timeout** exception that can be caught by the **catch** operation. The **timeout** exception is an emergency exit for cases where a called procedure neither replies nor raises an exception within a predetermined time (see clause 22.3.1).

**Catch any exception**

A **catch** operation with no argument list allows any valid exception to be caught. The most general case is without using the **from** keyword. *CatchAnyException* will also catch the **timeout** exception.

**Catch on any port**

To **catch** an exception on any port use the **any** keyword.

**Catch on any port from a port array**

To **catch** an exception on any port from a specific port array, indices use the **any from** *PortArrayRef*syntax where PortArrayRefshallbe areference to a port array identifier**.** It is also possible to store the index of a port in a single-dimensional port array at which the operation was successful to a variable of type integer or, in case of multi‑dimensional port arrays the index of the successful port to an integer array or record of integer variable. When checking the port array for matching exceptions, the port indices to be checked are iterated from lowest to highest. If the port array is multi-dimensional, then the ports are iterated over from innermost to outermost array dimension from lowest to highest index for each dimension, e.g. [0][0], [0][1], [1][0], [1][1]. The first port which matches all the criteria will cause the operation to be successful even if other ports in the array would also meet the criteria.

The catch on any port from a port array operation can not be used to catch a call timeout.

***Restrictions***

In addition to the general static rules of TTCN‑3 given in clause 5, the following restrictions apply:

1. The **catch** operation shall only be used at procedure-based ports. The type of the caught exception shall be specified in the signature of the procedure that raised the exception.
2. No binding of the incoming values to the terms of the expression or to the template shall occur. The assignment of the exception values to variables shall be made in the assignment part of the **catch** operation.
3. Catching **timeout** exceptions shall be restricted to the exception handling part of a call. No further matching criteria (including a **from** part) and no assignment part is allowed for a **catch** operation that handles a **timeout** exception.
4. Exception values accepted by *catch any exception* shall not be assigned to a variable, i.e. the **value** clause shall not be present.
5. If *CatchAnyException* is used in the response and exception handling part of a **call** operation, it shall only treat exceptions raised by the procedure invoked by the **call** operation.
6. *AddressRef* for retrieving the sending entity shall be of type **address**, **component** or of the type provided in the address declaration of the port type of the port instance referenced in the **catch** operation.
7. The *PortArrayRef* shall be a reference to a port array variable identifier.
8. The index redirection shall only be used when the operation is used on an any from port array construct.
9. If the index redirection is used for single-dimensional arrays, the type of the integer variable shall allow storing the highest index of the respective port array.
10. If the index redirection is used for multi-dimensional arrays, the size of the integer array or record of integer type shall exactly be the same as the dimension of the respective port array, and the its type shall allow storing the highest index (from all dimensions) of the port array.
11. If a variable referenced in the **value**, **sender** or **@index** clause is a lazy or fuzzy variable, the expression assigned to this variable is equal to the result produced by the **catch** operation, i.e. later evaluation of the lazy or fuzzy variable does not lead to repeated invocation of the **catch** operation.

***Examples***

EXAMPLE 1: Basic catch

MyPort.**catch**(MyProc, **integer:** MyVar); // Catches an integer exception of value

// MyVar raised by MyProc at port MyPort.

MyPort.**catch**(MyProc, MyVar); // Is an alternative to the previous example.

MyPort.**catch**(MyProc, A<B); // Catches a boolean exception

MyPort.**catch**(MyProc, MyType:{5, MyVar}); // In-line template definition of an exception value.

MyPort.**catch**(MyProc, **charstring**:"Hello")**from** MyPeer; // Catches "Hello" exception from MyPeer

EXAMPLE 2: Catch with storing value and/or sender in variables

MyPort.**catch**(MyProc, MyType:?) **from** MyPartner -> **value** MyVar;

// Catches an exception from MyPartner and assigns its value to MyVar.

MyPort.**catch**(MyProc, MyTemplate(5)) -> **value** MyVarTwo **sender** MyPeer;

// Catches an exception, assigns its value to MyVarTwo and retrieves the

// address of the sender.

MyPort.**catch**(MyProc, MyTemplate(5)) -> **value** (MyVarThree:= f1)

**sender** MyPeer;

// Catches an exception, assigns the value of its field f1 to MyVarThree and retrieves the

// address of the sender.

EXAMPLE 3: The Timeout exception

MyPort.**call**(MyProc:{5,MyVar}, 20E-3) {

[] MyPort.**getreply**(MyProc:{?, ?}) { }

[] MyPort.**catch**(**timeout**) { // timeout exception after 20ms

**setverdict**(**fail**);

**stop**;

}

}

EXAMPLE 4: Catch any exception

MyPort.**catch**;

MyPort.**catch** **from** MyPartner;

MyPort.**catch** -> **sender** MySenderVar;

EXAMPLE 5: Catch on any port

**any port**.**catch;**

EXAMPLE 6: Catch on any port from port array

**type** **port** MyPort **procedure** { **inout** MyProc }

**type** **component** MyComponent {

**port** MyPort p[10][10];

}

**var** **integer** i[2];

**any** **from** p.**catch**(MyProc, MyType:?) -> @**index** **value** i;

// Catching an incoming exception of type MyType on any port in the port array p and

// storing the index of the port on which the matching was successful first

#### A.1.6.4.2 Port operations

CommunicationStatements ::= [SendStatement](#TSendStatement) |

[CallStatement](#TCallStatement) |

[ReplyStatement](#TReplyStatement) |

[RaiseStatement](#TRaiseStatement) |

[ReceiveStatement](#TReceiveStatement) |

[TriggerStatement](#TTriggerStatement) |

[GetCallStatement](#TGetCallStatement) |

[GetReplyStatement](#TGetReplyStatement) |

[CatchStatement](#TCatchStatement) |

[CheckStatement](#TCheckStatement) |

[ClearStatement](#TClearStatement) |

[StartStatement](#TStartStatement) |

[StopStatement](#TStopStatement) |

[HaltStatement](#THaltStatement) |

[CheckStateStatement](#TCheckStateStatement)

SendStatement ::= [ArrayIdentifierRef](#TArrayIdentifierRef) [Dot](#TDot) [PortSendOp](#TPortSendOp)

PortSendOp ::= [SendOpKeyword](#TSendOpKeyword) "(" [InLineTemplate](#TInLineTemplate) ")" [[ToClause](#TToClause)]

SendOpKeyword ::= "send"

ToClause ::= [ToKeyword](#TToKeyword) ([InLineTemplate](#TInLineTemplate) |

[AddressRefList](#TAddressRefList) |

[AllKeyword](#TAllKeyword) [ComponentKeyword](#TComponentKeyword)

)

AddressRefList ::= "(" [InLineTemplate](#TInLineTemplate) {"," [InLineTemplate](#TInLineTemplate)} ")"

ToKeyword ::= "to"

CallStatement ::= [ArrayIdentifierRef](#TArrayIdentifierRef) [Dot](#TDot) [PortCallOp](#TPortCallOp) [[PortCallBody](#TPortCallBody)]

PortCallOp ::= [CallOpKeyword](#TCallOpKeyword) "(" [CallParameters](#TCallParameters) ")" [[ToClause](#TToClause)]

CallOpKeyword ::= "call"

CallParameters ::= [InLineTemplate](#TInLineTemplate) ["," [CallTimerValue](#TCallTimerValue)]

CallTimerValue ::= [Expression](#TExpression) | [NowaitKeyword](#TNowaitKeyword)

NowaitKeyword ::= "nowait"

PortCallBody ::= "{" [CallBodyStatementList](#TCallBodyStatementList) "}"

CallBodyStatementList ::= {[CallBodyStatement](#TCallBodyStatement) [[SemiColon](#TSemiColon)]}+

CallBodyStatement ::= [CallBodyGuard](#TCallBodyGuard) [StatementBlock](#TStatementBlock)

CallBodyGuard ::= [AltGuardChar](#TAltGuardChar) [CallBodyOps](#TCallBodyOps)

CallBodyOps ::= [GetReplyStatement](#TGetReplyStatement) | [CatchStatement](#TCatchStatement)

ReplyStatement ::= [ArrayIdentifierRef](#TArrayIdentifierRef) [Dot](#TDot) [PortReplyOp](#TPortReplyOp)

PortReplyOp ::= [ReplyKeyword](#TReplyKeyword) "(" [InLineTemplate](#TInLineTemplate) [[ReplyValue](#TReplyValue)] ")" [[ToClause](#TToClause)]

ReplyKeyword ::= "reply"

ReplyValue ::= [ValueKeyword](#TValueKeyword) [Expression](#TExpression)

RaiseStatement ::= [ArrayIdentifierRef](#TArrayIdentifierRef) [Dot](#TDot) [PortRaiseOp](#TPortRaiseOp)

PortRaiseOp ::= [RaiseKeyword](#TRaiseKeyword) "(" [Signature](#TSignature) "," [InLineTemplate](#TInLineTemplate) ")"

[[ToClause](#TToClause)]

RaiseKeyword ::= "raise"

ReceiveStatement ::= [PortOrAny](#TPortOrAny) [Dot](#TDot) [PortReceiveOp](#TPortReceiveOp)

PortOrAny ::= [ArrayIdentifierRef](#TArrayIdentifierRef) | ([AnyKeyword](#TAnyKeyword) ([PortKeyword](#TPortKeyword) | [FromKeyword](#TFromKeyword)

[VariableRef](#TVariableRef)))

PortReceiveOp ::= [ReceiveOpKeyword](#TReceiveOpKeyword) ["(" [InLineTemplate](#TInLineTemplate) ")"] [[FromClause](#TFromClause)]

[[PortRedirect](#TPortRedirect)]

ReceiveOpKeyword ::= "receive"

FromClause ::= [FromKeyword](#TFromKeyword) ([InLineTemplate](#TInLineTemplate) |

[AddressRefList](#TAddressRefList) |

[AnyKeyword](#TAnyKeyword) [ComponentKeyword](#TComponentKeyword)

)

FromKeyword ::= "from"

PortRedirect ::= [PortRedirectSymbol](#TPortRedirectSymbol) (([ValueSpec](#TValueSpec) [[SenderSpec](#TSenderSpec)] [[IndexSpec](#TIndexSpec)]) |

([SenderSpec](#TSenderSpec) [[IndexSpec](#TIndexSpec)]) |

[IndexSpec](#TIndexSpec)

)

PortRedirectSymbol ::= "->"

ValueSpec ::= [ValueKeyword](#TValueKeyword) ([VariableRef](#TVariableRef) | ("(" [SingleValueSpec](#TSingleValueSpec) {","

[SingleValueSpec](#TSingleValueSpec)}

")"))

SingleValueSpec ::= [VariableRef](#TVariableRef) [[AssignmentChar](#TAssignmentChar) [FieldReference](#TFieldReference) [ExtendedFieldReference](#TExtendedFieldReference)]

ShortValueSpec ::= [ValueKeyword](#TValueKeyword) [VariableRef](#TVariableRef)  
  
/\*STATIC SEMANTICS – FieldReference shall not be ParRef and ExtendedFieldReference shall not be TypeDefIdentifier\*/

ValueKeyword ::= "value"

SenderSpec ::= [SenderKeyword](#TSenderKeyword) [VariableRef](#TVariableRef)

SenderKeyword ::= "sender"

TriggerStatement ::= [PortOrAny](#TPortOrAny) [Dot](#TDot) [PortTriggerOp](#TPortTriggerOp)

PortTriggerOp ::= [TriggerOpKeyword](#TTriggerOpKeyword) ["(" [InLineTemplate](#TInLineTemplate) ")"] [[FromClause](#TFromClause)]

[[PortRedirect](#TPortRedirect)]

TriggerOpKeyword ::= "trigger"

GetCallStatement ::= [PortOrAny](#TPortOrAny) [Dot](#TDot) [PortGetCallOp](#TPortGetCallOp)

PortGetCallOp ::= [GetCallOpKeyword](#TGetCallOpKeyword) ["(" [InLineTemplate](#TInLineTemplate) ")"] [[FromClause](#TFromClause)]

[[PortRedirectWithParam](#TPortRedirectWithParam)]

GetCallOpKeyword ::= "getcall"

PortRedirectWithParam ::= [PortRedirectSymbol](#TPortRedirectSymbol) [RedirectWithParamSpec](#TRedirectWithParamSpec)

RedirectWithParamSpec ::= ([ParamSpec](#TParamSpec) [[SenderSpec](#TSenderSpec)] [[IndexSpec](#TIndexSpec)]) |

([SenderSpec](#TSenderSpec) [[IndexSpec](#TIndexSpec)]) |

[IndexSpec](#TIndexSpec)

ParamSpec ::= [ParamKeyword](#TParamKeyword) [ParamAssignmentList](#TParamAssignmentList)

ParamKeyword ::= "param"

ParamAssignmentList ::= "(" ([AssignmentList](#TAssignmentList) | [VariableList](#TVariableList)) ")"

AssignmentList ::= [VariableAssignment](#TVariableAssignment) {"," [VariableAssignment](#TVariableAssignment)}

VariableAssignment ::= [VariableRef](#TVariableRef) [AssignmentChar](#TAssignmentChar) [Identifier](#TIdentifier)

VariableList ::= [VariableEntry](#TVariableEntry) {"," [VariableEntry](#TVariableEntry)}

VariableEntry ::= [VariableRef](#TVariableRef) | [Minus](#TMinus)

GetReplyStatement ::= [PortOrAny](#TPortOrAny) [Dot](#TDot) [PortGetReplyOp](#TPortGetReplyOp)

PortGetReplyOp ::= [GetReplyOpKeyword](#TGetReplyOpKeyword) ["(" [InLineTemplate](#TInLineTemplate) [[ValueMatchSpec](#TValueMatchSpec)]

")"] [[FromClause](#TFromClause)] [[PortRedirectWithValueAndParam](#TPortRedirectWithValueAndParam)]

PortRedirectWithValueAndParam ::= [PortRedirectSymbol](#TPortRedirectSymbol) [RedirectWithValueAndParamSpec](#TRedirectWithValueAndParamSpec)

RedirectWithValueAndParamSpec ::= ([ShortValueSpec](#TShortValueSpec) [[ParamSpec](#TParamSpec)] [[SenderSpec](#TSenderSpec)]

[[IndexSpec](#TIndexSpec)]) | [RedirectWithParamSpec](#TRedirectWithParamSpec)

GetReplyOpKeyword ::= "getreply"

ValueMatchSpec ::= [ValueKeyword](#TValueKeyword) [InLineTemplate](#TInLineTemplate)

CheckStatement ::= [PortOrAny](#TPortOrAny) [Dot](#TDot) [PortCheckOp](#TPortCheckOp)

PortCheckOp ::= [CheckOpKeyword](#TCheckOpKeyword) ["(" [CheckParameter](#TCheckParameter) ")"]

CheckOpKeyword ::= "check"

CheckParameter ::= [CheckPortOpsPresent](#TCheckPortOpsPresent) |

[FromClausePresent](#TFromClausePresent) |

[RedirectPresent](#TRedirectPresent)

FromClausePresent ::= [FromClause](#TFromClause) [[PortRedirectSymbol](#TPortRedirectSymbol) (([SenderSpec](#TSenderSpec)

[[IndexSpec](#TIndexSpec)]) |

[IndexSpec](#TIndexSpec))]

RedirectPresent ::= [PortRedirectSymbol](#TPortRedirectSymbol) (([SenderSpec](#TSenderSpec) [[IndexSpec](#TIndexSpec)]) |

[IndexSpec](#TIndexSpec))

CheckPortOpsPresent ::= [PortReceiveOp](#TPortReceiveOp) |

[PortGetCallOp](#TPortGetCallOp) |

[PortGetReplyOp](#TPortGetReplyOp) |

[PortCatchOp](#TPortCatchOp)

CatchStatement ::= [PortOrAny](#TPortOrAny) [Dot](#TDot) [PortCatchOp](#TPortCatchOp)

PortCatchOp ::= [CatchOpKeyword](#TCatchOpKeyword) ["(" [CatchOpParameter](#TCatchOpParameter) ")"] [[FromClause](#TFromClause)]

[[PortCatchRedirect](#TPortCatchRedirect)]

PortCatchRedirect ::= [PortRedirectSymbol](#TPortRedirectSymbol) (([ShortValueSpec](#TShortValueSpec) [[SenderSpec](#TSenderSpec)] [[IndexSpec](#TIndexSpec)]) |

([SenderSpec](#TSenderSpec) [[IndexSpec](#TIndexSpec)]) |

[IndexSpec](#TIndexSpec)

)

CatchOpKeyword ::= "catch"

CatchOpParameter ::= [Signature](#TSignature) "," [InLineTemplate](#TInLineTemplate) | [TimeoutKeyword](#TTimeoutKeyword)

ClearStatement ::= [PortOrAll](#TPortOrAll) [Dot](#TDot) [ClearOpKeyword](#TClearOpKeyword)

PortOrAll ::= [ArrayIdentifierRef](#TArrayIdentifierRef) | [AllKeyword](#TAllKeyword) [PortKeyword](#TPortKeyword)

ClearOpKeyword ::= "clear"

StartStatement ::= [PortOrAll](#TPortOrAll) [Dot](#TDot) [StartKeyword](#TStartKeyword)

StopStatement ::= [PortOrAll](#TPortOrAll) [Dot](#TDot) [StopKeyword](#TStopKeyword)

StopKeyword ::= "stop"

HaltStatement ::= [PortOrAll](#TPortOrAll) [Dot](#TDot) [HaltKeyword](#THaltKeyword)

HaltKeyword ::= "halt"

AnyKeyword ::= "any"

CheckStateStatement ::= [PortOrAllAny](#TPortOrAllAny) [Dot](#TDot) [CheckStateKeyword](#TCheckStateKeyword) "(" [SingleExpression](#TSingleExpression)

")"

PortOrAllAny ::= [PortOrAll](#TPortOrAll) | [AnyKeyword](#TAnyKeyword) [PortKeyword](#TPortKeyword)

CheckStateKeyword ::= "checkstate"