

Java Runtime Exceptions

User's Guide

Version 3.4

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA, Inc. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with that agreement. No part of this guide may be reproduced or retransmitted in any form or by any means electronic, mechanical, or otherwise, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of BEA, Inc.

Copyright © 2001-2007 BEA, Inc. All rights reserved. All BEA Products are trademarks or registered trademarks of BEA, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

JAVA RUNTIME EXCEPTIONS	5
TRANSFORMEXCEPTION.....	7
FIELDS/CONTEXT PROPERTIES OF TRANSFORMEXCEPTION.....	11
CORE EXCEPTIONS	13
RUNTIME EXCEPTIONS.....	13
<i>TransformRuntimeException</i>	13
<i>FieldNotFoundException</i>	14
<i>TransformNullValueException</i>	15
VALIDATION EXCEPTIONS	16
<i>FieldValidationException</i>	17
<i>SectionConstraintException</i>	18
GENERAL EXCEPTIONS	18
<i>TransformSQLException</i>	19
<i>KeyGenerationException</i>	20
<i>FieldNullException</i>	20
<i>FieldParsingException</i>	21
<i>FieldTypeMismatchException</i>	21
PLUG-IN RELATED EXCEPTIONS	22
SWIFT PLUG-IN EXCEPTIONS	23
<i>SwiftParseException</i>	23
<i>SwiftTokenizeException</i>	24
<i>SwiftWriteException</i>	25
FIX PLUG-IN EXCEPTIONS.....	26
<i>FIXParsingException</i>	26
<i>FIXWriterException</i>	27
SIAC CANONICAL PLUG-IN EXCEPTIONS	27
<i>SIACCanonicalParsingException</i>	27
<i>SIACCanonicalWriterException</i>	28
FCS PLUG-IN EXCEPTIONS	28
<i>FCSParseException</i>	29
REAL PLUG-IN EXCEPTIONS.....	29
<i>RealParseException</i>	29
<i>RealWriteException</i>	30
XML PLUG-IN EXCEPTIONS	31
<i>XMLParseException</i>	31
<i>XMLWriteException</i>	31
UNIVERSAL PLUG-IN EXCEPTIONS	32
<i>UniversalParseException</i>	32
<i>UniversalWriteException</i>	33
ASCII DELIMITED PLUG-IN EXCEPTIONS	34
<i>ASCIIDelimitedParseException</i>	34
<i>ASCIIDelimitedTokenizeException</i>	34
<i>ASCIIDelimitedWriteException</i>	35

ASCII FIXED PLUG-IN EXCEPTIONS	36
<i>ASCIIFixedParseException</i>	36
<i>ASCIIFixedWriteException</i>	36
COMMAND PROCESSOR EXCEPTIONS.....	37
UNHANDLEDMESSAGEEXCEPTION	37
UNRECOGNIZEDMESSAGEEXCEPTION	38
INITIALIZATIONEXCEPTION	39

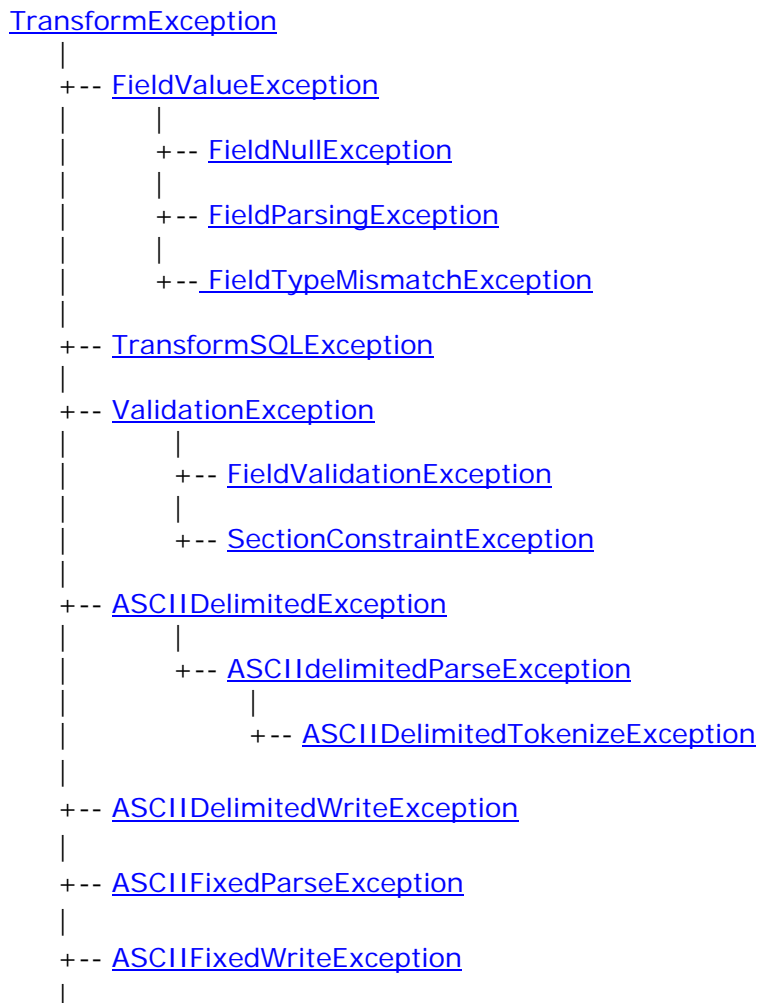
Java Runtime Exceptions

The exceptions thrown by Runtime environment system can be classified into the following categories:

1. [Core Exceptions](#)
2. [Plug-In Related Exceptions](#)
3. [Command Processor Exceptions](#)

While the core exceptions are common for all Plug-Ins, the Plug-In related exceptions are specific to the corresponding Plug-In. CP exceptions are thrown by input/output handlers during initialization/invoke. TransformException is the root of all exceptions thrown by the Runtime environment system except TransformRuntimeException that is derived from java.lang.RuntimeException.

The TransformException hierarchy is given below.



TransformException

```
+-- InitializationException
|
+-- UnhandledMessageException
|
|   +-- UnrecognizedMessageException
|
+-- FCSParseException
|
+-- FIXParsingException
|
+-- FIXWriterException
|
+-- KeyGenerationException
|
+-- RealParseException
|
+-- RealWriteException
|
+-- SIACCanonicalParsingException
|
+-- SIACCanonicalWriterException
|
+-- SwiftException
|
|   +-- SwiftParseException
|   |
|   |   +-- SwiftTokenizeException
|   |
|   +-- SwiftWriteException
|
+-- UniversalParseException
|
+-- UniversalwriteException
|
+-- XMLParseException
|
+-- XMLWriteException
```

TransformException

TransformException represents the root of all exceptions thrown by Runtime environment system except TransformRuntimeException. The caller can just catch TransformException or if finer details are needed any derived class can be caught.

The table given below summarizes the fields of TransformException.

Field Name	Mandatory/ Optional	Max Length	Description
Type	Mandatory	100	Type of the exception. Always 'TransformException'.
Message	Mandatory	500	Descriptive error message.
ErrorCode	Optional	100	Error code corresponding to the error that has occurred. In case of validation this is the user defined error code in the cartridge. In case of other errors, this is the error code that has been specified in message.properties.
Severity	Mandatory	100	Severity of the error that has occurred. Possible values include <ul style="list-style-type: none">fatalerrorwarn
Cascadable	Mandatory	N/A	Whether the exception is cascadable or not. Possible values include <ul style="list-style-type: none">truefalse
FieldName	Optional	100	Name of the field in which error has occurred. Present in case of validation errors, parsing/writing errors that occur while parsing/writing a field's value.

FieldID	Optional	100	Name of the field in which error has occurred. Present in case of validation errors.
Error-Code	Optional	100	Error code corresponding to the error that has occurred. This occurs only in case of TransformRuntime exceptions.
Error-Phase	Mandatory	100	Phase in which error has occurred. Allowed values are <ul style="list-style-type: none"> • Input • Output • Internal Message
Error-Type	Optional	100	Type of error that has occurred. Allowed values are <ul style="list-style-type: none"> • Parsing • Required • Validation • Input Mapping • Processing • Output Writing
Field-Value	Optional	10000	Value of the field in which error has occurred.
Location	Optional	100	Location in input/output message where the error has occurred. Exceptions thrown during input parsing phase of input records in batch mode (applicable for ASCII Delimited and XML formats) always include the 'Location' field. Allowed values are

			<ul style="list-style-type: none"> • Header • Record • Trailer
Error-Record	Optional	10000	<p>The entire record in which the field that has resulted in error is present.</p> <p>All the fields present in the record are displayed along with the field that has resulted in error.</p> <ul style="list-style-type: none"> • Exceptions thrown during the 'Internal Message' phase always include the 'Error-Record' field.
Error-Record-Index	Optional	100	<p>The index of the record in which the field that has resulted in error is present.</p> <p>Please note that the index starts from 0. This means that if error has occurred in a field present in the first record, then the index would be 0.</p> <p>Exceptions thrown during input parsing phase of input records in batch mode (applicable for ASCII Delimited and XML formats) always include the 'Error-Record-Index' field.</p>
Error-Line	Optional	1000	The actual line in the input message where error has occurred.
Internal-Code	Optional	100	Internal error code that corresponds to the actual error code.
line	Optional	20	Line number where the error has occurred.
column	Optional	20	<p>Position (column) in the line where the error has occurred.</p> <p>The line number is specified by the 'line' field.</p>

Index	Optional	100	Index where the error has occurred in the input data. The index is calculated from the beginning of data.
Trace	Optional	1000	Execution trace
subfield	Optional	100	Specific to SWIFT. Name of the subfield in which error has occurred.
Field	Optional	100	Specific to SWIFT. Tag for the field in which error has occurred.
sequence	Optional	100	Specific to SWIFT. Name of the sequence in which error has occurred.
qualifier	Optional	100	Specific to SWIFT. If error has occurred in a generic field, the name of the qualifier in which error has occurred.

Notes:

- All the fields of TransformException except the Cascadable field are of String type.
- Some of the error fields are not applicable in all formats (e.g. 'sequence' is specific to 'Swift').
- The error fields are applicable only in case of Java runtime errors.
- <StackTrace> is optional and it can be suppressed.

See Also:

[Fields/Context properties of TransformException](#)
[Java Runtime Exceptions](#)

Fields/Context properties of TransformException

The fields of TransformException are either defined as fields in this class or as a context property.

The table given below lists the TransformException fields that are defined as fields in the class.

Field Name	The TransformException method used to access the corresponding value
Type	getType()
Message	getMessage()
ErrorCode	GetErrorCode()
Severity	getSeverity()
Cascadable	getCascadable()
FieldName	GetFieldName()
FieldID	getFieldID()

The table given below lists the TransformException fields that are set as context properties of the exception object thrown. The values of these properties can be accessed using the `getContextProperty(java.lang.String name)` method of TransformException.

Field Name	Name of the Constant as defined in TransformException that can be used in the <code>getContextProperty(java.lang.String name)</code> method of TransformException to access the corresponding value
Error-Code	ERROR_CODE
Error-Phase	ERROR_PHASE
Error-Type	ERROR_TYPE
Field-Value	ERROR_FIELD_VALUE

Location	ERROR_LOCATION
Error-Record	ERROR_RECORD
Error-Record-Index	ERROR_RECORD_INDEX
Error-Line	ERROR_LINE
Internal-Code	INTERNAL_CODE
line	LINE
column	COLUMN
Index	INDEX
Trace	ERROR_TRACE

The table given below lists the TransformException fields that can also be accessed using the methods of TransformException even though they are set as context properties of the exception object thrown. See the API documentation for more details.

Field Name	The TransformException method used to access the corresponding value
Error-Phase	getErrorPhase()
Error-Type	getErrorType()
Field-Value	getFieldValue()

See Also:

[TransformException](#)

Core Exceptions

These exceptions are thrown by the core runtime system and these are applicable for all types of Plug-Ins.

The core exceptions can be further classified into the following categories:

1. [Runtime Exceptions](#)
2. [General Exceptions](#)
3. [Validation Exception](#)

See Also:

[Java Runtime Exceptions](#)

Runtime Exceptions

The runtime exceptions caused typically by programming errors are represented by TransformRuntimeException and its subclasses. These exceptions are thrown by the core runtime system and these are applicable for all types of Plug-Ins. Please note that the TransformRuntimeException class is derived from unchecked java.lang.RuntimeException.

The hierarchy of TransformRuntimeException is given below:

```
TransformRuntimeException
|
+-- FieldNotFoundException
|
+-- TransformNullValueException
```

See Also:

[Validation Exceptions](#)

[Core Exceptions](#)

TransformRuntimeException

This exception represents runtime exceptions caused by programming errors. Invalid input field values and invalid arguments passed to function calls used in formula also result in this kind of exception.

```
<Message>Unexpected exception. Date parsing error. '20031225' not in expected
format 'yyyy-MM-dd'</Message>
<ErrorCode>SRT563</ErrorCode>
<Severity>fatal</Severity>
```

```

<Cascadable>true</Cascadable>
<Error-Phase>Input</Error-Phase>
<Internal-Code>SRT563</Internal-Code>
<Error-Record>
  <?xml version="1.0" encoding="UTF-8" ?>
  <Data>
    <EncryptMethod>1</EncryptMethod>
    <HeartBtInt>2</HeartBtInt>
    <CustomDate>20031225</CustomDate>
  </Data>
</Error-Record>
<StackTrace>
Unexpected exception. Date parsing error. '20031225' not in expected format
'yyyy-MM-dd'
com.tplus.transform.runtime.TransformRuntimeException: Date parsing error.
'20031225' not in expected format 'yyyy-MM-dd'
...
</StackTrace>

```

Fields within TransformRuntimeException

- Message
- ErrorCode
- Severity
- Cascadable
- Error-Phase
- Internal-Code
- Error-Record
- StackTrace

See Also:

[Runtime Exceptions](#)
[FieldNotFoundException](#)

FieldNotFoundException

This exception can happen at any phase (input, Internal Message or output), in the following cases:

- using a field within a formula without checking the occurrence of its parent section trying to access a field which is not defined as part of the data object

```

<Type>TransformException</Type>
<Message>Unexpected runtime error. 'Field with name Account not
defined'.</Message>
<ErrorCode>SRT216</ErrorCode>
<Severity>fatal</Severity>
<Cascadable>true</Cascadable>
<Internal-Code>SRT216</Internal-Code>
<Error-Phase>Internal Message</Error-Phase>
<Error-Record>
    ...
</Error-Record>
<StackTrace>
    Unexpected runtime error. 'Field with name Account not defined'.
    com.tplus.transform.runtime.FieldNotFoundException: Field with name
Account not defined
    at
com.tplus.transform.runtime.DataObjectMetaInfo.getFieldMetaInfo(DataObjectMe
taInfo.java:168)
    ...
</StackTrace>

```

See Also:

[Runtime Exceptions](#)

TransformNullValueException

This is a subclass of TransformRuntimeException and it is thrown when trying to access an empty field (field with null value) from a formula. This can be avoided by checking for null values using the IsNull() and IsNotNull() functions.

```

<Message>Unexpected exception. Attempt to access field 'XPR' with null
value</Message>
<ErrorCode>SRT500</ErrorCode>
<Severity>fatal</Severity>
<Cascadable>true</Cascadable>
<Error-Phase>Input</Error-Phase>
<Internal-Code>SRT500</Internal-Code>
<Error-Record>
    <?xml version="1.0" encoding="UTF-8" ?>
    <Data>
        <ClOrdID>BHA 0066/11172003</ClOrdID>
        <HandlInst>1</HandlInst>
        <Symbol>SU</Symbol>
        <Side>1</Side>
        <OrderQty>1000</OrderQty>
    </Data>
</Error-Record>

```

```

        <OrdType>2</OrdType>
        <Price>21.76</Price>
        <TimeInForce>0</TimeInForce>
        <Rule80A>A</Rule80A>
    </Data>
</Error-Record>
<StackTrace>
Unexpected exception. Attempt to access field 'XPR' with null value
com.tplus.transform.runtime.TransformNullValueException: Attempt to access
field 'XPR' with null value
...
</StackTrace>

```

Fields within TransformNullValueException

- Message
- ErrorCode
- Severity
- Cascadable
- Error-Phase
- Internal-Code
- Error-Record
- StackTrace

See Also:

[Runtime Exceptions](#)

Validation Exceptions

The ValidationException class and its subclasses (shown in the hierarchy tree given below) represent errors while validating the message/field.

ValidationException

```

|
+-- FieldValidationException
|
+-- SectionConstraintException

```

See Also:

[Core Exceptions](#)

FieldValidationException

This exception can happen at any phase (input, Internal Message or output) and it exception is thrown when the validation rule applied for a field/message fails.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>0532 is not valid.</Message>
  <ErrorCode>T50</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>61.Entry_Date</FieldName>
  <FieldID>61[0].Entry_Date</FieldID>
  <Internal-Code>F61-D</Internal-Code>
  <Error-Type>Validation</Error-Type>
</TransformException>
```

Fields within FieldValidationException

The optional fields are marked with question mark (?).

- Type
- Message
- ErrorCode
- In case of SWIFT format, this field will contain the error code as defined in SWIFT SRG.
- Severity
- Cascadable
- Internal-Code
- FieldName
- This field is set to 'Message' if the 'Applies To' column of the corresponding Formula Validation is left empty.
- FieldID
- Please also note that this field is populated only when the validation rule is applied to a field. It will not be populated when the validation rule is specified at the message level, i.e. if the 'Applies To' column of the Formula Validation is left empty.
- Error-Type
- This is always set to 'Validation'.

See Also:

[Validation Exceptions](#)

SectionConstraintException

This exception can happen at any phase (input, Internal Message or output) and it is thrown when the number of elements in a section does not match the Min/Max Occurs properties (repeating/optional properties) specified for that section.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>The number of elements (3) in the section 'items.item' is
greater than 2.</Message>
  <ErrorCode>SRT302</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>items.item</FieldName>
  <FieldID>items.item</FieldID>
  <Internal-Code>SRT302</Internal-Code>
  <Error-Phase>Output</Error-Phase>
  <Error-Type>Output Writing</Error-Type>
</TransformException>
```

Fields within SectionConstraintException

The optional fields are marked with question mark (?).

- Type
- Message
- ErrorCode
- Severity
- Cascadable
- FieldName
- FieldID
- Internal-Code
- Error-Phase
- Error-Type

See Also:

[Validation Exceptions](#)

General Exceptions

TransformSQLException, KeyGenerationException and FieldValueException along with its subclasses belong to this category of exceptions.

- [TransformSQLException](#)
- [KeyGenerationException](#)

- FieldValueException
 - |
 - +-- [FieldNullException](#)
 - |
 - +-- [FieldParsingException](#)
 - |
 - +-- [FieldTypeMismatchException](#)

See Also:

[Core Exceptions](#)

TransformSQLException

Thrown by the Persistence Designer when one of the following operations fails: persisting NO, updating NO, removing NO and executing query (defined using Persistence Designer -> Queries UI).

```
<TransformException>
  <Type>TransformException</Type>
  <Message>
    Error persisting normalized object. SQL error : ORA-01401: inserted
    value too large for column
  </Message>
  <ErrorCode>SRT636</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>SRT636</Internal-Code>
  <Trace>at PersistInColumnsFlow.Persist1(Persist Invoice)</Trace>
</TransformException>
```

Fields within TransformSQLException

- Type
- Message
- ErrorCode
- Severity
- Cascadable
- Internal-Code
- Trace

See Also:

[General Exceptions](#)

KeyGenerationException

This exception is thrown in the following cases:

- if there is a problem in creating the connection to the specified datasource
- if the specified table is not found in the datasource
- if an SQL exception is thrown while executing database operations

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Error generating unique key. SQLException:
java.sql.SQLException: Table not found: UNIQUEKEYGENTBL in statement [select
CurrentKey from UniqueKeyGenTbl]</Message>
  <ErrorCode>SRT631</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>SRT631</Internal-Code>
</TransformException>
```

See Also:

[General Exceptions](#)

FieldNullException

After completing the post-processing of a normalized object, its mandatory fields are checked for the presence of their value. If a mandatory field is not assigned a value, it results in this exception.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Not-null check failed. The field 'TotalCost' has null
value</Message>
  <ErrorCode>SRT600</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>TotalCost</FieldName>
  <FieldID>TotalCost</FieldID>
  <Internal-Code>SRT600</Internal-Code>
  <Trace>at Input2NOFlow.Validate1(Validate)</Trace>
</TransformException>
```

Fields within FieldNullException

- Type
- Message

- ErrorCode
- Severity
- Cascadable
- FieldName
- FieldID
- Internal-Code
- Trace

See Also:

[General Exceptions](#)

FieldParsingException

This exception is thrown in the following cases:

- If a mandatory field/section is missing in the input message
- If a duplicate field/section is present in the input message
- If the value does not match the corresponding field type
- Incorrect value in case of FILLER fields in Universal format

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Field 'RecordID' cannot be null.</Message>
  <ErrorCode>SRT129</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>RecordID</FieldName>
  <Internal-Code>SRT129</Internal-Code>
  <Error-Type>Required</Error-Type>
  <Location>Record</Location>
  <Error-Phase>Input</Error-Phase>
</TransformException>
```

See Also:

[General Exceptions](#)

FieldTypeMismatchException

This exception occurs when a data access function (such as GetInt(), GetString(), etc. in the aggregate function category) is used in a formula to access a field of a data object (element of a section) whose type does not match the type expected by the data access function. This exception can happen at any phase (input, Internal Message or output).

```

<TransformException>
  <Type>TransformException</Type>
  <Message>
    Type mismatch while accesing field 'ItemID'. The field is not of
    specified type.
  </Message>
  <ErrorCode>SRT580</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>SRT580</Internal-Code>
  <Error-Phase>Internal Message</Error-Phase>
  <Error-Record>
    <?xml version="1.0" encoding="UTF-8" ?>
      <NewOrderBT>
        <Item>
          <ItemID>ITM1</ItemID>
          <Qty>5</Qty>
          <Price>100.0</Price>
        </Item>
      </NewOrderBT>
    </Error-Record>
</TransformException>

```

See Also:

[General Exceptions](#)

Plug-In Related Exceptions

The exceptions listed here apply to you only if you are using the corresponding plug in.

Plug-In related exceptions can be classified into the following:

- Parse Exceptions
- Write Exceptions

The parse exceptions happen at Input Parsing phase. At this time, the input object is not instantiated yet, so the <Error-Field-ID> will not be populated.

The write exceptions happen when the value of a mandatory field is missing (null) or the output value violates the specified constraint/format.

See Also:

[Swift Plug-In Exceptions](#)

[FIX Plug-In Exceptions](#)
[SIAC Canonical Plug-In Exceptions](#)
[FCS Plug-In Exceptions](#)
[REAL Plug-In Exceptions](#)
[XML Plug-In Exceptions](#)
[Universal Plug-In Exceptions](#)
[ASCII Delimited Plug-In Exceptions](#)
[ASCII Fixed Plug-In Exceptions](#)

Swift Plug-In Exceptions

The Swift Plug-In throws the following exceptions:

- [SwiftParseException](#)
- [SwiftTokenizeException](#)
- [SwiftWriteException](#)

See Also:

[Plug-In Related Exceptions](#)

SwiftParseException

Reasons for this exception include the following:

- Extra characters left at the end of a field/block/message.
- Missing mandatory field/block.
- Unable to locate end of a block.
- The field value violates the specified constraint such as length or format.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>In the value (1,123) of the subfield 'Amount', number of digits
following the comma, exceeds the maximum number (2) allowed for the specified
currency 'USD'.</Message>
  <ErrorCode>C03</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>B.32H.Amount</FieldName>
  <FieldID>B.32H.Amount</FieldID>
  <Internal-Code>SWT313</Internal-Code>
  <field>32</field>
  <Error-Line>:32H:USD1,123</Error-Line>
  <line>22</line>
  <column>14</column>
  <sequence>B</sequence>
```

```

        <subfield>Amount</subfield>
        <Field-Value>1,123</Field-Value>
        <Error-Phase>Input</Error-Phase>
        <Error-Type>Parsing</Error-Type>
    </TransformException>

```

See Also:

[Swift Plug-In Exceptions](#)

SwiftTokenizeException

This is a subclass of `SwiftParseException` and these exceptions are thrown when parsing the value corresponding to a sub-field.

Reasons for this exception include the following:

- One of the characters in the value corresponding to a subfield does not match the Swift format character used in the specifying the subfield format
- The value corresponding to a multi-line format exceeds the line limit.
- The integer part of decimal number is missing
- A decimal separator (comma) in the amount/number subfield is missing
- Multiple commas in a decimal number
- Sign character (N) expected for a subfield is missing
- Separator character expected for a subfield is missing
- Literal expected for a subfield is missing
- Unexpected additional characters at the end of a field
- Not enough characters found for a subfield
- Subfield itself is missing

```

<TransformException>
  <Type>TransformException</Type>
  <Message>A decimal separator (comma) in the amount/number subfield
  'Amount' with format '15d' is missing. Illegal value '0'.</Message>
  <ErrorCode>T43</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>A.32a.Amount</FieldName>
  <FieldID>A.32a.Amount</FieldID>
  <subfield>Amount</subfield>
  <field>32</field>
  <Error-Line>:32:NINR0</Error-Line>
  <line>6</line>
  <column>9</column>
  <Internal-Code>SWT310</Internal-Code>
  <Field-Value>0</Field-Value>

```



```

    <Error-Phase>Input</Error-Phase>
    <Error-Type>Parsing</Error-Type>
</TransformException>

```

See Also:

[Swift Plug-In Exceptions](#)

SwiftWriteException

Reasons for this exception include the following:

- Missing mandatory field/sub-field/qualifier.
- Repeating qualifier.
- Unexpected format option for swift field.
- Unexpected additional characters at the end of field.
- In the value of a sub-field, the number of digits following the comma exceeds the maximum number allowed for the specified currency.

```

<TransformException>
  <Type>TransformException</Type>
  <Message>Length of subfield 'Sender's_Reference' with format '16x' must
be less than or equal to 16 characters, found 17 characters. Illegal value
'12345678901234567'.</Message>
  <ErrorCode>T33</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>A.20.Sender's_Reference</FieldName>
  <FieldID>A.20.Sender's_Reference</FieldID>
  <subfield>Sender's_Reference</subfield>
  <Field-Value>12345678901234567</Field-Value>
  <Internal-Code>SWT303C</Internal-Code>
  <field>20</field>
  <sequence>A</sequence>
  <Error-Phase>Output</Error-Phase>
  <Error-Type>Output Writing</Error-Type>
</TransformException>

```

See Also:

[Swift Plug-In Exceptions](#)

FIX Plug-In Exceptions

The FIX Plug-In throws the following exceptions:

- [FIXParsingException](#)
- [FIXWriterException](#)

See Also:

[Plug-In Related Exceptions](#)

FIXParsingException

Reasons for this exception include the following:

- Blob type tag value pair has incorrect data length, character after the specified length is not the SOH character.
- Unexpected end of input while looking for data.
- Empty tag or value.
- Cannot convert tag to integer.
- Unexpected tag.
- Unexpected FIX data at the end.
- When the value corresponding to a Boolean/Boolean_4_1 FIX type field does not start with Y or N.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Unexpected tag 789</Message>
  <ErrorCode>FIX112</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>FIX112</Internal-Code>
  <Location>Header</Location>
  <Error-Phase>Input</Error-Phase>
  <Error-Type>Parsing</Error-Type>
</TransformException>
```

See Also:

[FIX Plug-In Exceptions](#)

FIXWriterException

This exception is thrown when a mandatory field is missing while writing the output.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>BeginInit is a mandatory field. Tag 'BeginInit[8]' cannot be
null.</Message>
  <ErrorCode>FIX114</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>FIX114</Internal-Code>
  <Error-Phase>Output</Error-Phase>
  <Error-Record>...</Error-Record>
</TransformException>
```

See Also:

[FIX Plug-In Exceptions](#)

SIAC Canonical Plug-In Exceptions

The SIAC Canonical Plug-In throws the following exceptions:

- [SIACCanonicalParsingException](#)
- [SIACCanonicalWriterException](#)

See Also:

[Plug-In Related Exceptions](#)

SIACCanonicalParsingException

Reasons for this exception include the following:

- Blob type tag value pair has incorrect data length, character after the specified length is not the SOH character.
- Unexpected end of input while looking for data
- Empty tag or value.
- Cannot convert tag to integer.
- Unexpected tag.
- Unexpected data at the end.
- When the value corresponding to a Boolean value does not start with Y or N.

```

<TransformException>
  <Type>TransformException</Type>
  <Message>Cannot convert tag to integer '1000 5'</Message>
  <ErrorCode>SIAC111</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>SIAC111</Internal-Code>
  <Error-Phase>Input</Error-Phase>
  <Error-Type>Parsing</Error-Type>
</TransformException>

```

See Also:

[SIAC Canonical Plug-In Exceptions](#)

SIACCanonicalWriterException

This exception is thrown when a mandatory field is missing while writing the output.

```

<TransformException>
  <Type>TransformException</Type>
  <Message>CFT_LimitPx is a mandatory field. Tag 'CFT_LimitPx[10677]'
cannot be null.</Message>
  <ErrorCode>SIAC115</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>SIAC115</Internal-Code>
  <Error-Phase>Output</Error-Phase>
  <Error-Record>
    ...
  </Error-Record>
</TransformException>

```

See Also:

[SIAC Canonical Plug-In Exceptions](#)

FCS Plug-In Exceptions

The FCS Plug-In throws the following exception:

- [FCSParseException](#)

See Also:

[Plug-In Related Exceptions](#)

FCSParseException

Reasons for this exception include the following:

- Input values does not conform to the specified format
- Unexpected characters at the end of line

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Expected field 'Exchange Code'</Message>
  <ErrorCode>FCS107</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>FCS107</Internal-Code>
  <line>2</line>
  <column>6</column>
  <Error-Line>ADMIN RPX XYZ</Error-Line>
  <Error-Phase>Input</Error-Phase>
  <Error-Type>Parsing</Error-Type>
</TransformException>
```

See Also:

[FCS Plug-In Exceptions](#)

REAL Plug-In Exceptions

The REAL Plug-In throws the following exceptions:

- [RealParseException](#)
- [RealWriteException](#)

See Also:

[Plug-In Related Exceptions](#)

RealParseException

Reasons for this exception while parsing REAL binary input:

- Unexpected end of input.
- Unexpected additional data at the end data layer.
- Unexpected additional data at the end of input.
- Unexpected section/field
- Missing mandatory field

Reasons for this exception while parsing REAL XML input:

- All exceptions thrown by the SAX parser.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Mandatory Field 'SWIFT Identification Of Security[00022213]' is
missing in the input.</Message>
  <ErrorCode>REAL102</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>REAL102</Internal-Code>
  <column>904</column>
  <Error-Phase>Input</Error-Phase>
  <Error-Type>Parsing</Error-Type>
</TransformException>
```

See Also:

[REAL Plug-In Exceptions](#)

RealWriteException

Reasons for this exception include the following:

- Length of the value corresponding to a header field does not correspond to the specified length.
- Missing mandatory field/section.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>The mandatory field 'SWIFT MT564 HEADER INFORMATION.RIAM
OID[00020450]' is missing.</Message>
  <ErrorCode>REAL113</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>REAL113</Internal-Code>
  <Error-Phase>Output</Error-Phase>
  <Error-Record>
    ...
  </Error-Record>
</TransformException>
```

See Also:

[REAL Plug-In Exceptions](#)

XML Plug-In Exceptions

The XML Plug-In throws the following exception:

- [XMLParseException](#)
- [XMLWriteException](#)

See Also:

[Plug-In Related Exceptions](#)

XMLParseException

Exceptions of this type include all exceptions thrown by the SAX parser.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Parsing Error. The entity "Site1" was referenced, but not
declared.</Message>
  <ErrorCode>XML101</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>XML101</Internal-Code>
  <line>4</line>
  <column>28</column>
</TransformException>
```

See Also:

[XML Plug-In Exceptions](#)

XMLWriteException

Reasons for this exception include the following:

- Missing mandatory field/section/attribute.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Missing mandatory section 'kunde.name'.</Message>
  <ErrorCode>SRT300</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>kunde.name</FieldName>
  <FieldID>kunde.name</FieldID>
  <Internal-Code>SRT300</Internal-Code>
```

```

    <Error-Phase>Output</Error-Phase>
    <Error-Type>Output Writing</Error-Type>
</TransformException>

```

See Also:

[XML Plug-In Exceptions](#)

Universal Plug-In Exceptions

The Universal Plug-In throws the following exceptions:

- [UniversalParseException](#)
- [UniversalWriteException](#)

See Also:

[Plug-In Related Exceptions](#)

UniversalParseException

Reasons for this exception include the following:

- Incorrect section/field tag
- Incorrect section/field tag separator
- Field value does not correspond to the specified format
- Unexpected additional data at the end of input.
- Not enough characters corresponding to a fixed length field
- Incorrect filler value
- Missing mandatory field

```

<TransformException>
  <Type>TransformException</Type>
  <Message>Section tag separator ':' expected. Found 'V'.</Message>
  <ErrorCode>UNV102</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>Tagged</FieldName>
  <Internal-Code>UNV102</Internal-Code>
  <Index>10</Index>
  <line>1</line>
  <column>11</column>
  <Error-Phase>Input</Error-Phase>
  <Error-Type>Parsing</Error-Type>
</TransformException>

```


See Also:

[Universal Plug-In Exceptions](#)

UniversalWriteException

Reasons for this exception include the following:

- Length of the value does not match the specified length
- Record length exceeding the specified upper limit
- Overflow of digits in packed decimal format
- More than one choice has a non-null value
- All the choices are null
- Missing mandatory field/section.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Cannot represent 9123 in 1 digits</Message>
  <ErrorCode>UNV126</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <FieldName>flt4</FieldName>
  <Internal-Code>UNV126</Internal-Code>
  <Error-Phase>Output</Error-Phase>
  <Error-Record>
    ...
  </Error-Record>
</TransformException>
```

See Also:

[Universal Plug-In Exceptions](#)

ASCII Delimited Plug-In Exceptions

The ASCII Delimited Plug-In throws the following exceptions:

ASCIIDelimitedException

```
|
+-- ASCIIDelimitedParseException
    |
    +-- ASCIIDelimitedTokenizeException
```

See Also:

[Plug-In Related Exceptions](#)

ASCIIDelimitedParseException

Reasons for this exception include the following:

- The input does not have the minimum number (two) of records
- Incorrect number of fields in header, record or trailer

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Incorrect number of fields in RecordData. Expected fields 2,
found 1</Message>
  <ErrorCode>ASC122</ErrorCode>
  <Severity>error</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>ASC122</Internal-Code>
  <Location>Record</Location>
  <Error-Phase>Input</Error-Phase>
  <Trace>at Flow2.Parse1(Parse Msg)
    at BuggyDynamicBinaryParamsInvocationFlow.Invoke2(Invoke Flows)</Trace>
</TransformException>
```

See Also:

[ASCII Delimited Plug-In Exceptions](#)

ASCIIDelimitedTokenizeException

Reasons for this exception include the following:

- In a quoted field the closing quote corresponding to the opening quote is not found
- Improper quoted token

- EOL reached before closing quote found

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Improper quoted token</Message>
  <ErrorCode>ASC119</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>ASC119</Internal-Code>
  <column>7</column>
  <line>2</line>
  <Error-Line>ITM1,"" ,10,100.0</Error-Line>
  <Trace>at CommaDelTestFlow.Parse1(Parse)</Trace>
</TransformException>
```

See Also:

[ASCII Delimited Plug-In Exceptions](#)

ASCIIDelimitedWriteException

Reasons for this exception include the following:

- Incorrect number of fields in header, record or trailer
- Missing mandatory field.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>The mandatory field 'joindate' is missing.</Message>
  <ErrorCode>ASC121</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>ASC121</Internal-Code>
  <Error-Phase>Output</Error-Phase>
  <Error-Record>
    ...
  </Error-Record>
</TransformException>
```

See Also:

[ASCII Delimited Plug-In Exceptions](#)

ASCII Fixed Plug-In Exceptions

The ASCII Fixed Plug-In throws the following exceptions:

- [ASCIIFixedParseException](#)
- [ASCIIFixedWriteException](#)

See Also:

[Plug-In Related Exceptions](#)

ASCIIFixedParseException

Reasons for this exception include the following:

- Unexpected additional data at the end of input.
- Not enough characters corresponding to a fixed length field
- Missing mandatory field

```
<TransformException>
  <Type>TransformException</Type>
  <Message>Unexpected characters at the end of sequence.</Message>
  <ErrorCode>ASCFIX104</ErrorCode>
  <Severity>fatal</Severity>
  <Cascadable>true</Cascadable>
  <Internal-Code>ASCFIX104</Internal-Code>
  <Index>13</Index>
  <line>1</line>
  <column>14</column>
</TransformException>
```

See Also:

[ASCII Fixed Plug-In Exceptions](#)

ASCIIFixedWriteException

Reasons for this exception include the following:

- Length of the value does not match the specified length
- Missing mandatory field/section.

```
<TransformException>
  <Type>TransformException</Type>
  <Message>The mandatory field 'Price' is missing.</Message>
  <ErrorCode>ASCFIX105</ErrorCode>
```

```

    <Severity>fatal</Severity>
    <Cascadable>true</Cascadable>
    <FieldName>Price</FieldName>
    <Internal-Code>ASCFIX105</Internal-Code>
    <Error-Phase>Output</Error-Phase>
    <Error-Record>
    ...
    </Error-Record>
</TransformException>

```

See Also:

[ASCII Fixed Plug-In Exceptions](#)

Command Processor Exceptions

These exceptions are thrown by input/output handlers of CP during initialization/invocation. The exception are,

[UnhandledMessageException](#)

[UnrecognizedMessageException](#)

[InitializationException](#)

See Also:

[Java Runtime Exceptions](#)

UnhandledMessageException

This exception is thrown by the input message router when none of the message routers matched the message.

```

<TransformException>
  <Message>None of the message routers matched the message
  <SwiftHeader>
    <MessageFormat>SWIFT</MessageFormat>
    <MessageType>103</MessageType>
    <Application_Identifier>F</Application_Identifier>
    <Service_Identifier>01</Service_Identifier>
    <LT_Identifier>PASOBEB0AXXX</LT_Identifier>
    <Session_Number>0230</Session_Number>
    <Sequence_Number>023948</Sequence_Number>
    <Input_Output_Identifier>O</Input_Output_Identifier>
    <Message_Type>103</Message_Type>
    <Message_Priority>N</Message_Priority>
  </SwiftHeader>

```

```

    </Message>
    <StackTrace>
        com.tplus.transform.runtime.external.UnhandledMessageException: None of
the message routers matched the message
        ...
    </StackTrace>
</TransformException>

```

Fields within UnhandledMessageException

- Message
- StackTrace

See Also:

[Command Processor Exceptions](#)

UnrecognizedMessageException

This exception is thrown when none of the message identifier(s) specified could handle the message.

```

<TransformException>
<Message>Unable to identify message. None of the message identifier(s)
specified could handle the message.</Message>
<StackTrace>
com.tplus.transform.runtime.external.UnrecognizedMessageException: Unable to
identify message. None of the message identifier(s) specified could handle the
message.
    at
com.tplus.transform.runtime.external.AbstractMessageRouter.identifyMessage(Abst
ractMessageRouter.java:68)
    at
com.tplus.transform.runtime.external.AbstractMessageRouter.processMessage(Abstr
actMessageRouter.java:100)
    ...
</StackTrace>
</TransformException>

```

Fields within UnrecognizedMessageException

- Message
- StackTrace

See Also:

[Command Processor Exceptions](#)

InitializationException

This exception is thrown by input/output handlers when there are problems in initializing them based on the configuration specified in the configuration files. This exception is caused mostly by improper or missing specification of a configuration property/setting.

Reasons for this exception include the following:

- When the output handler is trying to instantiating the specified the device writer class, if it results in error.
- Not setting the 'output.writer' property for the RMI output handler.
- Not setting the 'output.device' and 'output.writer' properties for the Queue output handler.
- Not setting the 'input.directory' property for a file input handler or specifying an input directory that does not exist.
- Specifying an illegal polling time value (the 'input.polling.time' property) for a file input handler.
- Not setting the 'output.directory' property for a file output writer or specifying an output directory that does not exist.
- Specifying an empty file name pattern (the 'file.name' property) for the file output writer.
- Specifying an illegal file mode value (the 'file.mode' property) for the file output writer.
- Referring to a message identifier from an <InputDef> tag (using the 'message.identifiers' property), that is not defined in the CP configuration file.
- If message routing configuration is defined without specifying any message identifier(s).
- Neither the 'input.format' property nor the 'internal.message' property is specified as part of the transform context of routing configuration.
- Specifying an unsupported appia interface (the 'appia.interface' property) for the FIX input/output handler.
- When the FIX input handler is trying to launch appia, if it results in error.
- When the input message router is trying to instantiate the specified error handler class.
- When the Queue input/output handler is trying to initialize the specified queue, if it results in error because of the following possible causes:
 - Unable to lookup/create the specified queue connection factory
 - Neither 'queue.adapter' nor 'queue.connectionfactory.name' property is specified.

Fields within InitializationException

- Message
- StackTrace

See Also:

[Command Processor Exceptions](#)