Java Runtime Exceptions

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

1. Core Exceptions
2. Plug-In Related Exceptions

While the core exceptions are common for all Plug-Ins, the Plug-In related exceptions are specific to the corresponding Plug-In. TransformException is the root of all exceptions thrown by the Runtime system except TransformRuntimeException that is derived from java.lang.RuntimeException.

The TransformException hierarchy is given below.

```
TransformException
 |--- FieldValueException
 |   |--- FieldNullException
 |   |--- FieldParsingException
 |   |--- FieldTypeMismatchException
 |--- TransformSQLException
 |--- ValidationException
 |   |--- FieldValidationException
 |   |--- SectionConstraintException
 |--- ASCIIDelimitedException
 |   |--- ASCIIdelimitedParseException
 |     |--- ASCIIDelimitedTokenizeException
 |--- ASCIIDelimitedWriteException
 |--- ASCIIFixedParseException
 |--- ASCIIFixedWriteException
TransformException
 |--- FCSException
```
TransformException

TransformException represents the root of all exceptions thrown by Runtime 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.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mandatory/Optional</th>
<th>Max Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Mandatory</td>
<td>100</td>
<td>Type of the exception. Always ‘TransformException’.</td>
</tr>
<tr>
<td>Message</td>
<td>Mandatory</td>
<td>500</td>
<td>Descriptive error message.</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>Optional</td>
<td>100</td>
<td>Error code corresponding to the error that has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In case of validation this is the user defined error code in the cartridge.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In case of other errors, this is the error code that has been specified in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>message.properties.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Severity</td>
<td>Mandatory</td>
<td>100</td>
<td>Severity of the error that has occurred. Possible values include</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- fatal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- warn</td>
</tr>
<tr>
<td>Cascadable</td>
<td>Mandatory</td>
<td>N/A</td>
<td>Whether the exception is cascadable or not. Possible values include</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- false</td>
</tr>
<tr>
<td>FieldName</td>
<td>Optional</td>
<td>100</td>
<td>Name of the field in which error has occurred. Present in case of validation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>errors, parsing/writing errors that occur while parsing/writing a field’s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value.</td>
</tr>
<tr>
<td>FieldID</td>
<td>Optional</td>
<td>100</td>
<td>Name of the field in which error has occurred. Present in case of validation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>errors.</td>
</tr>
<tr>
<td>Error-Code</td>
<td>Optional</td>
<td>100</td>
<td>Error code corresponding to the error that has occurred. This occurs only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in case of TransformRuntime exceptions.</td>
</tr>
<tr>
<td>Error-Phase</td>
<td>Mandatory</td>
<td>100</td>
<td>Phase in which error has occurred. Allowed values are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Internal Message</td>
</tr>
<tr>
<td>Error-Type</td>
<td>Optional</td>
<td>100</td>
<td>Type of error that has occurred. Allowed values are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Parsing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Validation</td>
</tr>
<tr>
<td>Field-Value</td>
<td>Optional</td>
<td>10000</td>
<td>Value of the field in which error has occurred.</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Location</td>
<td>Optional</td>
<td>100</td>
<td>Location in input/output message where the error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Exceptions thrown during input parsing phase of input records in batch mode (applicable for ASCII Delimited and XML formats) always include the ‘Location’ field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Allowed values are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Header</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Record</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Trailer</td>
</tr>
<tr>
<td>Error-Record</td>
<td>Optional</td>
<td>10000</td>
<td>The entire record in which the field that has resulted in error is present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All the fields present in the record are displayed along with the field that has resulted in error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Exceptions thrown during the ‘Internal Message’ phase always include the ‘Error-Record’ field.</td>
</tr>
<tr>
<td>Error-Record-Index</td>
<td>Optional</td>
<td>100</td>
<td>The index of the record in which the field that has resulted in error is present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Exceptions thrown during input parsing</td>
</tr>
<tr>
<td>Field</td>
<td>Required</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>phase of input records in batch mode (applicable for ASCII Delimited and XML formats) always include the ‘Error-Record-Index’ field.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error-Line</td>
<td>Optional</td>
<td>1000</td>
<td>The actual line in the input message where error has occurred.</td>
</tr>
<tr>
<td>Internal-Code</td>
<td>Optional</td>
<td>100</td>
<td>Internal error code that corresponds to the actual error code.</td>
</tr>
<tr>
<td>line</td>
<td>Optional</td>
<td>20</td>
<td>Line number where the error has occurred.</td>
</tr>
<tr>
<td>column</td>
<td>Optional</td>
<td>20</td>
<td>Position (column) in the line where the error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The line number is specified by the 'line' field.</td>
</tr>
<tr>
<td>Index</td>
<td>Optional</td>
<td>100</td>
<td>Index where the error has occurred in the input data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The index is calculated from the beginning of data.</td>
</tr>
<tr>
<td>Trace</td>
<td>Optional</td>
<td>1000</td>
<td>Execution trace</td>
</tr>
<tr>
<td>subfield</td>
<td>Optional</td>
<td>100</td>
<td>Specific to SWIFT. Name of the subfield in which error has occurred.</td>
</tr>
<tr>
<td>Field</td>
<td>Optional</td>
<td>100</td>
<td>Specific to SWIFT. Tag for the field in which error has occurred.</td>
</tr>
<tr>
<td>sequence</td>
<td>Optional</td>
<td>100</td>
<td>Specific to SWIFT. Name of the sequence in which error has occurred.</td>
</tr>
<tr>
<td>qualifier</td>
<td>Optional</td>
<td>100</td>
<td>Specific to SWIFT. If error has occurred in a generic field, the name of the qualifier in which error has occurred.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>The TransformException method used to access the corresponding value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>getType()</td>
</tr>
<tr>
<td>Message</td>
<td>getMessage()</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>GetErrorCode()</td>
</tr>
<tr>
<td>Severity</td>
<td>getSeverity()</td>
</tr>
<tr>
<td>Cascadable</td>
<td>getCascadable()</td>
</tr>
<tr>
<td>FieldName</td>
<td>GetFieldName()</td>
</tr>
<tr>
<td>FieldID</td>
<td>getFieldID()</td>
</tr>
</tbody>
</table>

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`.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Name of the Constant as defined in TransformException that can used in the <code>getContextProperty(java.lang.String name)</code> method of <code>TransformException</code> to access the corresponding value</th>
</tr>
</thead>
</table>

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.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>The TransformException method used to access the corresponding value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error-Phase</td>
<td>getErrorPhase()</td>
</tr>
<tr>
<td>Error-Type</td>
<td>getErrorType()</td>
</tr>
<tr>
<td>Field-Value</td>
<td>fieldValue()</td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Error-Record</td>
<td></td>
</tr>
<tr>
<td>Error-Record-Index</td>
<td></td>
</tr>
<tr>
<td>Error-Line</td>
<td></td>
</tr>
<tr>
<td>Internal-Code</td>
<td></td>
</tr>
<tr>
<td>line</td>
<td>LINE</td>
</tr>
<tr>
<td>column</td>
<td>COLUMN</td>
</tr>
<tr>
<td>Index</td>
<td>INDEX</td>
</tr>
<tr>
<td>Trace</td>
<td>ERROR_TRACE</td>
</tr>
</tbody>
</table>

**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>
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(DataObjectMetaInfo.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 (?).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type</td>
</tr>
<tr>
<td>Message</td>
<td>Message</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>ErrorCode</td>
</tr>
<tr>
<td>Severity</td>
<td>Severity</td>
</tr>
<tr>
<td>Cascadable</td>
<td>Cascadable</td>
</tr>
<tr>
<td>FieldName</td>
<td>FieldName</td>
</tr>
<tr>
<td>FieldID</td>
<td>FieldID</td>
</tr>
<tr>
<td>Internal-Code</td>
<td>Internal-Code</td>
</tr>
<tr>
<td>Error-Phase</td>
<td>Error-Phase</td>
</tr>
<tr>
<td>Error-Type</td>
<td>Error-Type</td>
</tr>
</tbody>
</table>

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
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

```xml
<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 accessing field 'ItemID'. The field is not of the 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](#)
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>
```
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

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.

See Also:
Swift Plug-In Exceptions

FIX Plug-In Exceptions

The FIX Plug-In throws the following exceptions:
**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>BeginString is a mandatory field. Tag 'BeginString[8]' cannot be null.</Message>
  <ErrorCode>FIX114</ErrorCode>
  <Severity>fatal</Severity>
</TransformException>
```
<Cascadable>true</Cascadable>
<Internal-Code>FIX114</Internal-Code>
<Error-Phase>Output</Error-Phase>
<Error-Record>...</Error-Record>
</TransformException>

See Also:
FIX Plug-In Exceptions
Plug-In Related 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
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.

```xml
<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.

```xml
<TransformException>
    <Type>TransformException</Type>
    <Message>Missing mandatory section 'kunde.name'.</Message>
    <ErrorCode>SRT300</ErrorCode>
</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>
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.

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

See Also:
ASCII Fixed Plug-In Exceptions
Java Runtime Exceptions