Sterling Standards Library

Standards Library Property Files

Version 6.0

Sterling Commerce
An IBM Company

Contents

Overview

Property files contain properties that control the operation of the application. By modifying the settings of these properties, you can customize the application, if necessary, to suit your business needs. Property files are located in the install_dir/properties directory and are usually named in the following manner: filename.properties. Some files have other suffixes including .xml, .xsl, .cfg, and .ini. Substitute the appropriate suffix for properties when needed in the instructions provided.

Caution: Since property files directly affect the operation of the application, please ensure that you fully understand the impact of property file changes. When changing configuration files, be sure that you have a complete backup of your system and have fully tested the changes in a test or development environment before moving the changes into production. For more information about working with any property file or property, please contact Sterling Commerce Customer Support.

Leading or trailing whitespace in property files will be respected by the application. This may cause a problem if the system is not expecting whitespace. When editing property files, be careful to trim leading and trailing whitespace before saving each file.

Overriding or Editing Property File Settings

The application supports the use of a customer override (edit) property file to override default property settings in the property files. The customer override property file is not changed during installation of the application upgrades or patches. To prevent having your customized settings overwritten, use overrides whenever possible rather than editing the property files.

If you have made changes to property files either directly or by editing the associated .in files in a previous version, your changes may be overwritten when a patch is applied. To prevent this, create a customer_overrides.properties file and reapply your modifications using overrides to the applicable property files in the customer_overrides.properties file.

For example, if you want to change the maximum size of interchange for ASC X12 in enveloping.properties to 100, you need to add the following line to the customer_overrides.properties file (and then this value can be overridden by setting the size directly in the envelope):

```
enveloping.enveloping.X12.MaxInterchangeSize=100
```

In this example, enveloping represents the enveloping.properties file, enveloping.X12 represents the ASC X12 standard, and MaxInterchangeSize represents the property configuring maximum size allowed for interchanges.

ach.properties

The ach.properties file is used to set record format definitions for ACH deenveloping. These parameters include all default settings for the enveloping user interface.

Set treatRequiredFieldsAsOptional=YES to specify that required fields (required by the ACH standards definition) should be treated as optional. By default, these fields are treated as mandatory.

The field format default is:

```
name,offset,length
```

The date format default is:

```
Date_YYMMDD, Date_MMDD, Date_MMYY
```

Configuration Settings

The following table describes properties used to configure the ach.properties file in the application:

Property	Description
deenvelope.ACH.File_Header_Record.field.[field_number]	Sets the File_Header_Record field number specified to a name, offset, length, type, and field requirement. **Example** `deenvelope.ACH.File_Header_Record.field.01=RecordTypeCode,0,1,Fixed=1,M`
deenvelope.ACH.template.01.field.[field_number]	Sets the template field number specified to a name, offset, length, type, and field requirement. **Example** `deenvelope.ACH.template.01.field.01=RecordTypeCode,0,1,Fixed=9,M`
deenvelope.ACH.[trans_type].file_control_record	Sets the file_control_record for the specified translation type. **Example** `deenvelope.ACH.ACK.file_control_record=01`
deenvelope.ACH.[trans_type].batch_header_record	Sets the batch_header_record for the specified translation type. **Example** `deenvelope.ACH.ADV.batch_header_record=06`
deenvelope.ACH.[trans_type].batch_control_record	Sets the batch_control_record for the specified translation type. **Example** `deenvelope.ACH.ARC.batch_control_record=11`
deenvelope.ACH.[trans_type].entry_detail_record	Sets the entry_detail_record for the specified translation type. **Example** `deenvelope.ACH.ATX.entry_detail_record=31`
deenvelope.ACH.[trans_type].addenda_record	Sets the addenda_record for the specified translation type. **Example** `deenvelope.ACH.CBR.addenda_record=37`

Property	Description
min_addenda_count.[trans_type]	Sets the minimum addenda record count for the specified translation type. **Example** `min_addenda_count.ACK=0`
max_addenda_count.[trans_type]	Sets the maximum addenda record count for the specified translation type. **Example** `max_addenda_count.BOC=1`
have_PRI	Defines SEC codes that have an addenda record with PRI. **Example** `have_PRI=CCD,CIE,CTX,DNE,ENR,PPD,TRX,WEB`
edirule.TotalAmount	Defines the total amount. **Example** `edirule.TotalAmount=EntryDetailRecord,TotalAmount, //TotalAmount/text(),NumberEquality,*,*,*`
edirule.RDFI	Defines the receiving DFI Identification. **Example** `edirule.RDFI=EntryDetailRecord, ReceivingDFIIdentification,// ReceivingDFIIdentification/text()://CheckDigit /text(),StringEquality,*,*,*`
edirule.DFIAccountNum	Defines the DFI account number. **Example** `edirule.DFIAccountNum=EntryDetailRecord, DFIAccountNumber,//DFIAccountNumber/text(), StringEquality,*,*,*`
edirule.SettlementDate	Defines the settlement date. **Example** `edirule.SettlementDate=BatchHeaderRecord, SettlementDate,//SettlementDate,JulianYYMMDDEquality ,*,*,*`
edirule.CreditDebitFlag	Sets the default credit/debit flag. **Example** `edirule.CreditDebitFlag=EntryDetailRecord, TransactionCode,//CreditDebitFlag/text(),Mapping,X12 ,820,*`
edirule.AccountQualifier	Defines the account qualifiers that are allowed. **Example** `edirule.AccountQualifier=EntryDetailRecord, TransactionCode,//AccountNumberQualifier/text(), Mapping,X12,820,*`

Property	Description
edirulemapping.CreditDebitFlag.[number]	Defines the default rule mapping for the credit/debit flag. **Example** `edirulemapping.CreditDebitFlag.22=C`
edirulemapping.AccountQualifier.[number]	Defines the default rule mapping for the account qualifier. **Example** `edirulemapping.AccountQualifier.53=LN`

chips.properties

The chips.properties file is used to set global configuration parameters and allowable values for the CHIPS standard.

Configuration Settings

The following table describes properties used to configure the chips.properties file in the application:

Property	Description

tag.[tag_number]	Defines the allowed CHIPS tags.
	Examples
	`tag.014=Valid Service Message Retrieval`
	`tag.016=Receive Retrieval Response`
	`tag.017=Resolver Retrieval Response`
	`tag.018=Invalid Enquiry/Retrieval`
	`tag.025=Payment Message Stored`
	`tag.027=Valid Service Message Response`
	`tag.028=Invalid Service Message Response`
	`tag.031=Receive`

transactioncode.[trans_code_number]	Defines the allowed CHIPS transaction codes.
	Examples
	`transactioncode.01=Delete`
	`transactioncode.02=Payment Preference`
	`transactioncode.08=Participant Enquiry`
	`transactioncode.10=Payment Message`
	`transactioncode.22=Service Message`
	`transactioncode.39=Payment Message Retrieval`
	`transactioncode.40=Receive Retrieval`

responsecode.[response_code_number]	Defines the allowed CHIPS response codes.
	Examples
	`responsecode.25=Payment Message Response`
	`responsecode.27=Service Message Response`
	`responsecode.29=Payment Preference Response`
	`responsecode.31=Receive Notification`

response.[response_number]	Defines the allowed CHIPS responses.
	Examples
	`response.25025=Payment Message Response`
	`response.27027=Valid Service Message Response`
	`response.27028=Invalid Service Message Response`
	`response.29050=Payment Preference Response`
	`response.29051=Invalid Payment Preference Response`
	`response.31031=Receive Notification`

Property	Description
RESPONSEHEADERFIELDS	Defines fields that must contain a response header. **Example** `RESPONSEHEADERFIELDS=011 014 016 017 018 025 027 028 031 036 038 039 040 050 051 057 058 068 069 101 102`
[REGULAR EXPRESSIONS]	Regular expressions for parsing a CHIPS message. **Example** `TRANCODEREGEX=^(\\d{2})`

enveloping.properties

The enveloping.properties file is used to set global configuration parameters for enveloping. These parameters include all the default setting for the enveloping user interface.

Note: This file should not be edited. Override property settings, if needed, using the customer_overrides.properties file. For detailed instructions on overriding property file settings, see the documentation for the customer_overrides.properties file.

Configuration Settings

The following table describes properties used to configure the enveloping.properties file in the application:

Property	Description

enveloping.[standard]	Specifies the business process used to apply an envelope to one or more messages and then uses the envelope data to process and translate them.
	Examples
	`enveloping.X12=X12EnvelopeUnified`
	`enveloping.EDIFACT=EDIFACTEnvelopeUnified`
	`enveloping.CII=CIIEnvelope`
	`enveloping.TRADACOMS=TRADACOMSEnvelope`
	`enveloping.ACH=ACHEnvelope`
	`enveloping.VDA=VDAEnvelope`
	`enveloping.SWIFT=SWIFTEnvelope`
	`enveloping.RND=RNDEnvelope`
	`enveloping.CHIPS=CHIPSEnvelope`
	`enveloping.FEDWIRE=FedwireEnvelope`

enveloping.inner.[standard]	Defines the inner envelope types for the specified standard that uses generic enveloping.
	Examples
	`enveloping.inner.TRADACOMS=MHD MTR`
	`enveloping.inner.VDA=VDA`
	`enveloping.inner.SWIFT=FIN`
	`enveloping.inner.RND=RND`
	`enveloping.inner.CHIPS=CHIPS`
	`enveloping.inner.FEDWIRE=FEDWIRE`

enveloping.outer.[standard]	Defines the outer envelope types for the specified standard that uses generic enveloping.
	Examples
	`enveloping.outer.TRADACOMS=STX END`
	`enveloping.outer.SWIFT=SWIFT FIN COPY`

Property	Description

enveloping.envelope_level.[standard].[envelope_level]	Defines the level of a generic envelope.
	Examples
	`enveloping.envelope_level.RND.RND=Interchange`
	`enveloping.envelope_level.SWIFT.FIN=Interchange`
	`enveloping.envelope_level.TRADACOMS.MHD MTR=Transaction`
	`enveloping.envelope_level.TRADACOMS.STX END=Interchange`
	`enveloping.envelope_level.VDA.VDA=Interchange`
	`enveloping.envelope_level.CHIPS.CHIPS=Interchange`
	`enveloping.envelope_level.FEDWIRE.FEDWIRE=Interchange`

enveloping.out_required.[standard]	Defines whether a standard using generic enveloping requires an outer envelope. Valid values are True and False (default). **Example** `enveloping.outer_required.TRADACOMS=TRUE`
deenveloping.support_swift_096	Specifies whether FIN Copy is supported for deenveloping. Valid values are True (default) and False.
enveloping.outbound_pd_values.[standard]	Specifies outbound envelope values to put into process data before the translation map is invoked, so that the values are available for validation during translation. **Example** `enveloping.outbound_pd_values.SWIFT=SenderID,ReceiverID,ValidationFlag`
enveloping.verify_addresses_while_deenveloping.[standard]_INBOUND	For standards that use generic enveloping, indicates that the Generic Deenvelope service should verify addresses in process data prior to running the deenvelope map. Valid values are False (default) and True. **Note:** Currently SWIFT Deenveloping map is the only map that may use this function, but the functionality is disabled by default. **Example** `enveloping.verify_addresses_while_deenveloping.SWIFT_FIN_INBOUND=TRUE`
enveloping.ack_processing_map.[standard]	Defines the map that is executed after deenveloping, if it is determined that a message is an acknowledgement. **Example** `enveloping.ack_processing_map.SWIFT=SWIFT_FIN_ACK`
enveloping.ack_generation_map.[standard]	Defines the map to use when generating an acknowledgement. **Example** `enveloping.ack_generation_map.SWIFT=SWIFT_FIN_UAK`
enveloping.inbound_filter_parameters.[standard]	Specifies additional parameters to be used in the envelope to match, as part of the inbound envelope filter. **Note:** No parameters are currently used by default. **Example** `enveloping.inbound_filter_parameters.SWIFT=ValidationFlag`

Property	Description
enveloping.inbound_post_filter_parameters.[standard]	Specifies additional parameters to be used in the envelope to match after the envelope filter is applied. **Note:** This property is used for backward compatibility when a new matching parameter is added, when envelopes exists that do not use the parameter. **Example** `enveloping.inbound_post_filter_parameters.SWIFT=ValidationFlag,EnvelopeFormat,\MessageFormat,APCOrFIN,ClosedUserGroup,SenderIDType,\ ReceiverIDType,MessageCategory`
enveloping.inbound_post_filter_parameter_defaults.[standard]	Specifies the default value to use for the envelope if a post-filter parameter is not defined. **Example** `enveloping.inbound_post_filter_parameter_defaults.SWIFT=*,FIN,\ *,*,*,*,*,Message`

enveloping.correl_override_default.[standard]	Defines whether a standard allows wildcard overrides on outbound envelopes. Valid values are: No, wildcardonly, or all.
	Note: If an entry does not exist in the properties file for a standard, it is assumed that the value is set to **No**.
	Examples
	`enveloping.correl_override_default.ACH=All`
	`enveloping.correl_override_default.CHIPS=No`
	`enveloping.correl_override_default.EDIFACT=All`
	`enveloping.correl_override_default.FEDWIRE=No`
	`enveloping.correl_override_default.RND=No`
	`enveloping.correl_override_default.SWIFT=No`
	`enveloping.correl_override_default.TRADACOMS=No`
	`enveloping.correl_override_default.VDA=No`
	`enveloping.correl_override_default.X12=All`

enveloping.genctrlnumname.nameVariations.[standard].inner	Specifies the total number of name variations to attempt when generating the control number name for standards using generic enveloping (that is, the total number of primary names and backup names that are specified in the user interface for the standard). **Example** `enveloping.genctrlnumname.nameVariations.SWIFT.inner=3`

Property	Description
enveloping.genctrlnumname.component.[standard].inner	Specifies the properties that are used to build the control number name. Allowable properties are: SenderID, SenderIDQual, ReceiverID, and ReceiverIDQual (all four parameters are pulled from the EDI state and work correctly with wildcards), and any envelope parameter. **Note:** In the enveloping user interface, the variable names must be set as the component name with the variation number appended; prepended by the type of name and an underscore. For example, genctrolnumname_SenderID1). **Example** `enveloping.genctrlnumname.component.SWIFT.inner=genctrlnumname_SenderID,genctrlnumname_SenderIDQual,genctrlnumname_ReceiverID,genctrlnumname_ReceiverIDQual,genctrlnumname_MessageFormat,genctrlnumname_BusinessArea,genctrlnumname_MessageType,genctrlnumname_ValidationFlag,genctrlnumname_Variant,genctrlnumname_Version`
enveloping.genctrlnumname.appendComponents.[standard].inner	Specifies the subset the components that appear at the end of the generated name (for example, Inbound or Outbound). **Note:** If a parameter is not specified for this property, the envelope type is appended by default. **Example** `enveloping.genctrlnumname.appendComponents.SWIFT.inner=genctrlnumname_EnvelopeFormat`
enveloping.genctrlnumname.optionalComponents.[standards].inner	Specifies a comma-separated list of optional components. **Example** `enveloping.genctrlnumname.optionalComponents.SWIFT.inner=genctrlnumname_SenderID,genctrlnumname_SenderIDQual,genctrlnumname_ReceiverID,genctrlnumname_ReceiverIDQual,genctrlnumname_MessageFormat,genctrlnumname_BusinessArea,genctrlnumname_MessageType,genctrlnumname_ValidationFlag,genctrlnumname_Variant,genctrlnumname_Version,genctrlnumname_EnvelopeFormat`

enveloping.genmapname.namevariations.[standard].inner	Defines dynamically generated names for the pre-enveloping translation map, including a count of the number of variations that the user interface allows.
	Example
	`enveloping.genmapname.nameVariations.SWIFT.inner=3`
	`enveloping.genmapname.component.SWIFT.inner=genmapname_SenderID,genmapname_SenderIDQual,genmapname_ReceiverID,genmapname_ReceiverIDQual,genmapname_MessageFormat,genmapname_BusinessArea,genmapname_MessageType,genmapname_ValidationFlag,genmapname_Variant,genmapname_Version`

Property	Description

enveloping.genmapname.component.[standard].inner	Defines dynamically generated names for the pre-enveloping translation map, including a list of components.
	Example
	`enveloping.genmapname.nameVariations.SWIFT.inner=3`
	`enveloping.genmapname.component.SWIFT.inner=genmapname_SenderID,genmapname_SenderIDQual,genmapname_ReceiverID,genmapname_ReceiverIDQual,genmapname_MessageFormat,genmapname_BusinessArea,genmapname_MessageType,genmapname_ValidationFlag,genmapname_Variant,genmapname_Version`

enveloping.genmapname.appendComponents.[standard].inner	Defines dynamically generated names for the pre-enveloping translation map, including a list of components that may be appended to build the name. **Example** `enveloping.genmapname.appendComponents.SWIFT.inner=genmapname_EnvelopeFormat`
enveloping.genmapname.optionalComponents.[standard].inner	Defines dynamically generated names for the pre-enveloping translation map, including a list of optional components that may be used to build the name. **Example** `enveloping.genmapname.optionalComponents.SWIFT.inner=genmapname_SenderID,genmapname_SenderIDQual,genmapname_ReceiverID,genmapname_ReceiverIDQual,genmapname_MessageFormat,genmapname_BusinessArea,genmapname_MessageType,genmapname_ValidationFlag,genmapname_Variant,genmapname_Version,genmapname_EnvelopeFormat`
enveloping.genbpname.namevariations.[standard].inner	Defines dynamically generated names for the business process, including a count of the number of variations that the user interface allows. **Example** `enveloping.genbpname.nameVariations.SWIFT.inner=3`
enveloping.genbpname.component.[standard].inner	Defines dynamically generated names for the business process, including a list of components. **Example** `enveloping.genbpname.component.SWIFT.inner=genbpname_SenderID,genbpname_SenderIDQual,genbpname_ReceiverID,genbpname_ReceiverIDQual,genbpname_MessageFormat,genbpname_BusinessArea,genbpname_MessageType,genbpname_ValidationFlag,genbpname_Variant,genbpname_Version`
enveloping.genbpname.appendComponents.[standard].inner	Defines dynamically generated names for the business process, including a list of components that may be appended to build the name. **Example** `enveloping.genbpname.appendComponents.SWIFT.inner=genbpname_EnvelopeFormat`

Property	Description
enveloping.genbpname. optionalComponents.[standard].inner	Defines dynamically generated names for the business process, including a list of optional components that may be used to build the name. **Example** `enveloping.genbpname.optionalComponents.SWIFT.inner=genbpname_SenderID,genbpname_SenderIDQual,genbpname_ReceiverID,genbpname_ReceiverIDQual,genbpname_MessageFormat,genbpname_BusinessArea,genbpname_MessageType,genbpname_ValidationFlag,genbpname_Variant,genbpname_Version,genbpname_EnvelopeFormat`
enveloping.generrorbpname. namevariations.[standard].inner	Defines dynamically generated names for the error business process, including a count of the number of variations that the user interface allows. **Example** `enveloping.generrorbpname.nameVariations.SWIFT.inner=3`
enveloping.generrorbpname.component. [standard].inner	Defines dynamically generated names for the error business process, including a list of components. **Example** `enveloping.generrorbpname.component.SWIFT.inner=generrorbpname_SenderID,generrorbpname_SenderIDQual,generrorbpname_ReceiverID,generrorbpname_ReceiverIDQual,generrorbpname_MessageFormat,generrorbpname_BusinessArea,generrorbpname_MessageType,generrorbpname_ValidationFlag,generrorbpname_Variant,generrorbpname_Version`
enveloping.generrorbpname. appendComponents.[standard].inner	Defines dynamically generated names for the error business process, including a list of components that may be appended to build the name. **Example** `enveloping.generrorbpname.appendComponents.SWIFT.inner=generrorbpname_EnvelopeFormat`
enveloping.generrorbpname. optionalComponents.[standard].inner	Defines dynamically generated names for the error business process, including a list of optional components that may be used to build the name. **Example** `enveloping.generrorbpname.optionalComponents.SWIFT.inner=generrorbpname_SenderID,generrorbpname_SenderIDQual,generrorbpname_ReceiverID,generrorbpname_ReceiverIDQual,generrorbpname_MessageFormat,generrorbpname_BusinessArea,generrorbpname_MessageType,generrorbpname_ValidationFlag,generrorbpname_Variant,generrorbpname_Version,generrorbpname_EnvelopeFormat`

Property	Description

enveloping.do.whitespace_processing.[SWIFT]	Flag indicating if whitespace between EDI segments should be removed for a specified standard. Valid values are: True and False.
	Examples
	`enveloping.do_edi_whitespace_processing.SWIFT=FALSE`
	`enveloping.do_edi_whitespace_processing.TRADACOMS=TRUE`
	`enveloping.do_edi_whitespace_processing.VDA=FALSE`
	`enveloping.do_edi_whitespace_processing.RND=FALSE`

deenveloping.[standard].ErrorOn[ack]ReconcileFailure	Flag indicating whether or not to error the Deenveloping service when acknowledgement reconciliation fails for a specified standard.
	Examples
	`deenveloping.X12.ErrorOn997ReconcileFailure=FALSE`
	`deenveloping.EDIFACT.ErrorOnCONTRLReconcileFailure=FALSE`
	`deenveloping.ACH.ErrorOnACKReconcileFailure=FALSE`
	`deenveloping.ErrorOnACKReconcileFailure=FALSE`

enveloping.encode_by_outer_envelope.standards	Indicates which standards use the outer envelope for EDI encoding (for batching purposes). **Note:** If a standard is added to this property, it must also be added to the **use_new_key_convention** property. The property value is a comma-separated list of standard names. Each standard in the list should also have a set of envelopes (by which to group) defined. If a document has the first type of **group_by_ envelope** defined, it will be grouped based on that envelope. Otherwise, the second type of envelope will be checked, and so on. **Examples** `enveloping.encode_by_outer_envelope.standards=ACH,EDIFACT,X12`

enveloping.group_by_envelope	Indicates which envelope should be used to group documents for batching purposes.
	Note: If a document has the first type of **group_by_ envelope** defined, it will be grouped based on that envelope. Otherwise, the second type of envelope will be checked, and so on.
	Examples
	`enveloping.group_by_envelope.ACH.1=ACH BATCH`
	`enveloping.group_by_envelope.X12.1=GS GE`
	`enveloping.group_by_envelope.EDIFACT.1=UNG UNE Syntax 4`

Property	Description
enveloping.use_new_key_convention.standards	Specifies a list of standards for which the multi-group key convention should be used (for example, DOC/GROUP-1_DOC1 instead of DOC/DOC-1). Standards in this list use the new key convention even if encoding by outer envelope is disabled for the **enveloping.encode_by_outer_envelope.standards** parameter. **Example** `enveloping.use_new_key_convention.standards=ACH,EDIFACT,X12`

enveloping.[standard].MaxInterchangeSize	Globally configures a size limit for transactions for the specified standard. This limit can be overridden by setting the maximum size directly in the envelope.
	Note: To set the size globally, remove the # before the standard for which you want to set the size, and set the maximum size after the equal sign.
	Example
	`# enveloping.X12.MaxInterchangeSize=0`
	`# enveloping.VDA.MaxInterchangeSize=0`
	`# enveloping.EDIFACT.MaxInterchangeSize=0`
	`# enveloping.TRADACOMS.MaxInterchangeSize=0`

enveloping.[standard].[transaction].MaxInterchangeSize	Globally configures a size limit for interchanges for the specified standard. This limit can be overridden by setting the maximum size directly in the envelope.
	Note: To set the size globally, remove the # before the standard/transaction for which you want to set the size, and set the maximum size after the equals sign.
	Examples
	`# enveloping.X12.810.MaxInterchangeSize=0`
	`# enveloping.X12.850.MaxInterchangeSize=0`
	`# enveloping.EDIFACT.ORDERS.MaxInterchangeSize=0`
	`#`
	`enveloping.TRADACOMS.INVOICES.MaxInterchangeSize=0`

enveloping.interchange_override_correlations.[standard]	List of correlations that override interchange envelope properties. **Note:** Documents must have the same values for all of these correlations to be placed in the same interchange. Always include EnvelopingInterchangeID in this list. If the standard allows for control number overrides, Out_InterchangeControlNumber should also be in this list. **Example** `enveloping.interchange_override_correlations.ACH=EnvelopingInterchangeID,Out_DestinationIdentification,\Out_OriginIdentification,Out_DestinationName,Out_OriginName,Out_ReferenceCode,Out_InterchangeControlNumber`

Property	Description
enveloping.group_override_correlations.[standard]	List of correlations that override group envelope properties. **Note:** Documents must have the same values for all of these correlations to be placed in the same group. Always include EnvelopingGroupID in this list. If the standard allows for control number overrides, Out_GroupControlNumber should also be in this list. **Example** `enveloping.group_override_correlations.ACH=EnvelopingGroupOrder,\Out_RDFI,Out_ODFI,Out_CompanyID,Out_DFIAccountNumber,Out_CompanyEntryDescription,\Out_CompanyName,Out_CompanyDescriptiveDate,Out_CompanyDiscretionaryData,\Out_DiscretionaryData,Out_TransactionCode,Out_IdentificationNumber,\Out_ReceivingCompanyIDNumber,Out_ForeignExchangeReference,Out_ISODestinationCountryCode,\Out_ISOOriginatingCurrencyCode,Out_ISODestinationCurrencyCode,Out_ACHOperatorData,\Out_RoutingNumberOfACHOperator,Out_ServiceClassCode,Out_EffectiveEntryDate,\Out_OriginatorStatusCode,Out_MessageAuthenticationCode,Out_ForeignExchangeIndicator,\Out_ForeignExchangeReferenceIndicator`
enveloping.group_dependency_values.[standard].1	List of parameter names that must be the same in a particular envelope for the documents to be grouped together. This is used, for example, when the group envelope uses values from the transaction envelope when building the output document. **Note:** The format is envelope type, parameter name, override correlation name (optional). **Example** `enveloping.group_dependency_values.EDIFACT.1=UNH UNT Syntax 4,MessageType,Out_MessageType`

enveloping.interchange_dependency_values.[standard].1	List of parameter names that must be the same in a particular envelope for the documents to be placed in the same interchange. This is used, for example, when the interchange envelope uses values from the transaction envelope when building the output document.
	Note: The format is envelope type, parameter name, override correlation name (optional). For EDIFACT, documents cannot be grouped together unless they have the same security settings.
	Examples
	`enveloping.interchange_dependency_values.ACH.1=ACH BATCH,BatchingKey`
	`enveloping.interchange_dependency_values.EDIFACT.1=UNG UNE Syntax 4,SecureGroupOutbound,Out_SecureGroupOutbound`

enveloping.docRepair.[standard]	Flag indicating whether document repair is enabled. Valid values are: True or False. **Note:** Only supported for SWIFT standard. **Example** `enveloping.docRepair.SWIFT=true`

Property	Description
enveloping.docRepair.[standard].EDIT.mailto	Specifies an e-mail address to which errors that occur on editing documents (during document repair) should be mailed. **Note:** Only supported for SWIFT standard. **Example** `enveloping.docRepair.SWIFT.EDIT.mailTo=&SI_ADMIN_MAIL_ADDR`
enveloping.docRepair.[standard].RESEND.mailto	Specifies an mail address to which errors that occur on resending documents (during document repair) should be mailed. **Note:** Only supported for SWIFT standard. **Example** `enveloping.docRepair.SWIFT.RESEND.mailTo=&SI_ADMIN_MAIL_ADDR`
enveloping.SWIFT.nextversion	For SWIFT only, specifies the next version for the patch/release. **Example** `enveloping.SWIFT.nextVersion=2006`
enveloping.SWIFT.nextversionStart	For SWIFT only, specifies the date (YYYYMMDD format) when the next version starts. **Example** `enveloping.SWIFT.nextVersionStart=20061118`
enveloping.SWIFT.currentversion	For SWIFT only, specifies the current version for the patch/release. **Example** `enveloping.SWIFT.currentVersion=2005`
enveloping.global.ui.maxControlNumberSize	Defines the default maximum control number size in the enveloping user interface. **Example** `enveloping.global.ui.maxControlNumberSize=16`
enveloping.maxTransactionControlNumberSize.[standard]	Defines the maximum size for transaction control numbers for the specified standard. **Example** `enveloping.maxTransactionControlNumberSize.SWIFT=9999999999999999`
enveloping.maxInterchangeControlNumberSize.[standard]	Defines the maximum size for interchange-level control numbers for the specified standard. **Example** `enveloping.maxInterchangeControlNumberSize.SWIFT=9999999999999999`
deenveloping.interchangetypes	Specifies the number of interchange types for deenveloping. **Example** `deenveloping.interchangetypes=13`
deenveloping.interchange.[standard]	Specifies the deenveloping interchanges for the indicated standard. **Example** `deenveloping.interchange.X12=x-application/edi-x12;0;ISA;IEA;variable;3;105;0;X12DeenvelopeUnified`

Property	Description
deenveloping.[standard]	Defines the deenveloping service used to deenvelope interchanges for the specified standard. **Example** `deenveloping.CII=CIIDeenvelope`
unpersistedStorageType	Storage type for unpersisted documents. Valid values are: FS (file system) or DB (database). **Example** `unpersistedStorageType=FS`
deenveloping.[standard].SequenceCheckRetryTimeout	Default period of time for retrying sequence checking. **Example** `deenveloping.X12.SequenceCheckRetryTimeout=30`
checkForBINSegment	For ASC X12 only, flag specifying whether to check for the BIN segment. Valid values are: True (default) and False. **Example** `checkForBINSegment=true`
replaceXMLChars	For EDIFACT only, indicates the default characters to be replaced in XML. **Example** `replaceXMLChars=,!@#$% ^&*+=(){}[]<>/;:`
trimEnvelopeParms	For JDBC Oracle databases only, whether to trim envelope parameters. Valid values are: True (default) and False. **Example** `trimEnvelopeParms=true`
deenveloping.ACH.IgnorePadRecords	For ACH only, whether to ignore record padding. Valid values are: True and False (default). **Example** `deenveloping.ACH.IgnorePadRecords=false`
usePostProcessorBPLinks	Whether to use post processor business process links. Valid values are: True and False (default). **Example** `usePostProcessorBPLinks=false`

Property	Description
EDIENVELOPE_MAX_LOCK_TIME	Specifies the maximum amount of time that the EDI Enveloping service will hold a lock on an enveloping process. This parameter enables you to fine tune your EDI processing. You can set this parameter in envelope.properties.in, and then use the same parameter here to override the properties file setting where necessary. The default value is 86400 seconds. This parameter is used in Deferred mode to avoid situations where enveloping service releases the lock before it is done processing a message-possibly due to slow response from the database. If there are concurrent EDI enveloping business processes running, an EDI Enveloping service in another business process could possibly retrieve the same message from the database and process it, which would result in duplicate outbound messages. **Example** `EDIENVELOPE_MAX_LOCK_TIME=86400`
EDIENVELOPE_LOCK_WAIT_TIME	Specifies the maximum amount of time that the EDI Enveloping service will wait for a lock before returning an error and reporting failure to obtain the lock. The service would use this parameter value when another EDI Enveloping business process has the lock, which causes this EDI Enveloping service instance to have to wait for the lock. The default value is 86400 seconds. This parameter is used in Deferred mode only. **Example** `EDIENVELOPE_LOCK_WAIT_TIME=86000`
ensureCorrelationUpdate	For database iSeries only, specifies whether to update correlations. Valid values are: Yes (default) and No. **Example** `ensureCorrelationUpdate=yes`
TRACKING_LEVEL	Specifies the level of tracking that is retrieved for the envelope when it is processed. Full specifies to track all events for the envelope.

requiredCorrelation.[correlation_number]	Defines the correlations required by the generic enveloping, deferred enveloping, and acknowledgement generation functionality.
	Examples
	`requiredCorrelation.1=InterchangeAuthInfo`
	`requiredCorrelation.2=InterchangeAuthInfoQualifier`
	`requiredCorrelation.3=InterchangeReceiverIDQualifier`
	`requiredCorrelation.4=InterchangeSecAuthInfo`

defaultHIPAAValidationLevel	Defines the default HIPAA validation level. **Example** `defaultHIPAAValidationLevel=4`

Property	Description
deenveloping.harness_hook_pd_values_interchange.[standard]	For generic standards, specifies which values in process data (after the break map is executed) are passed on to the deenvelope data harness hook. **Example** `deenveloping.harness_hook_pd_values_interchange.RND=MessageType,MessageVersion,\InterchangeControlNumber,InterchangeDateTime,SenderID,ReceiverID,InternalTransmitterCode,\InternalReceiverCode,SenderName,ReceiverName,TestFlag,HeaderMessageType,Standard`
encrypt.parameters	List of envelope parameter names whose corresponding values # should be encrypted when stored in the database (for example, for password or encryption key values). **Example** `encrypt.parameters=KeyPart1,KeyPart2`
deenveloping.MAX_INTERCHANGE_COUNT	Specifies the maximum number of non-EDI Interchanges that can be processed in a file. The default is 500 and an error is generated when this value is exceeded. **Example** `deenveloping.MAX_INTERCHANGE_COUNT=500`
document.lifespan	Specifies the document lifespan (in minutes) when an acknowledgement for an EDI Outbound document is expected. The outbound workflow is not archived/purged unitl the value specified if the corresponding acknowledgement is not received. The default value corresponds to 30 days. **Example** `document.lifespan=30`
runErropBPonEnvelopeNotFound	Specifies whether the Interchange Error business process will run when UNH envelope is not found. The default is false (the business process will not run). **Example** `runErropBPonEnvelopeNotFound=false`

Property	Description
enveloping.X12.MaxDocsPer Interchange	This property is commented out by default (not used). If you use ASC X12 enveloping and remove the # from the property, and specify a value greater than zero, it allows you to specify the Maximum number of documents per interchange. If you do not specify a value or if you specify zero, the system will not check for maximum documents per interchange. If you specify a value greater than zero, only that number of documents (or less) will include in an interchange. This parameter can work in conjunction with the **Limit Interchange Size** parameter to limit the number of documents per interchange and also limit the size of the interchange. This may result in less than the maximum number of documents depending upon the size limit you impose. **Note:** Note that anything specified in the envelope definition will override the global value specified in the enveloping.properties file. Additionally, any correlation overrides will override the value specified in the envelope definition and the enveloping.properties file.
enveloping.EDIFACT.MaxDocsPer Interchange	This property is commented out by default (not used). If you use EDIFACT enveloping and remove the # from the property, and specify a value greater than zero, it allows you to specify the Maximum number of documents per interchange. If you do not specify a value or if you specify zero, the system will not check for maximum documents per interchange. If you specify a value greater than zero, only that number of documents (or less) will include in an interchange. This parameter can work in conjunction with the **Limit Interchange Size** parameter to limit the number of documents per interchange and also limit the size of the interchange. This may result in less than the maximum number of documents depending upon the size limit you impose. **Note:** Note that anything specified in the envelope definition will override the global value specified in the enveloping.properties file. Additionally, any correlation overrides will override the value specified in the envelope definition and the enveloping.properties file.

Property	Description
enveloping.VDA.MaxDocsPer Interchange	This property is commented out by default (not used). If you use VDA enveloping and remove the # from the property, and specify a value greater than zero, it allows you to specify the Maximum number of documents per interchange. If you do not specify a value or if you specify zero, the system will not check for maximum documents per interchange. If you specify a value greater than zero, only that number of documents (or less) will include in an interchange. This parameter can work in conjunction with the **Limit Interchange Size** parameter to limit the number of documents per interchange and also limit the size of the interchange. This may result in less than the maximum number of documents depending upon the size limit you impose. **Note:** Note that anything specified in the envelope definition will override the global value specified in the enveloping.properties file. Additionally, any correlation overrides will override the value specified in the envelope definition and the enveloping.properties file.
enveloping.TRADACOMS.MaxDocsPer Interchange	This property is commented out by default (not used). If you use TRADACOMS enveloping and remove the # from the property, and specify a value greater than zero, it allows you to specify the Maximum number of documents per interchange. If you do not specify a value or if you specify zero, the system will not check for maximum documents per interchange. If you specify a value greater than zero, only that number of documents (or less) will include in an interchange. This parameter can work in conjunction with the **Limit Interchange Size** parameter to limit the number of documents per interchange and also limit the size of the interchange. This may result in less than the maximum number of documents depending upon the size limit you impose. **Note:** Note that anything specified in the envelope definition will override the global value specified in the enveloping.properties file. Additionally, any correlation overrides will override the value specified in the envelope definition and the enveloping.properties file.

einvoicing.properties

The einvoicing.properties file is used to set global configuration parameters for the application. These parameters include all the default settings for the Integrated Archive user interface as well as default settings for the e-Invoice services.

Note: This file should not be edited. Override property settings, if needed, using the customer_overrides.properties file. For detailed instructions on overriding property file settings, see the documentation for the customer_overrides.properties file.

Configuration Settings

The following table describes properties used to configure the einvoicing.properties file in the application:

Property	Description
ExternalArchive_BP_Name	Specifies name of the business process used to process external archive files by the application services. **Example** `ExternalArchive_BP_Name=eInvoiceExternalArchive`
Admin_Email_Address	Defines the administrative E-mail address used by the application services to send notifications. **Example** `Admin_Email_Address=&SI_ADMIN_MAIL_ADDR`
trustweaver.sig.1	Defines the TrustWeaver electronic signature. **Note:** If you modify this value, a Signature Mismatch Error will be posted by the TrustWeaver Request Service. **Example** `trustweaver.sig.1=rO0ABXQABkRFU2VkZXVyAAJbQqzzF/gGCFTgAgAAeHAAAAAIbuyt6ZsEgUZzcgARamF2YS5sYW5nLkludGVnZXI4qCk94GHOAIAAUkABXZhbHVleHIAEGphdmEubGFuZy5OdW1iZXKGrJUdC5TgiwIAAHhwAAAAZXVxAH4AAQAAAAi07plz6ejUQA==`
unitSizeKB	Defines the unit count that is sent to TrustWeaver. The size of the documents (in KB) that are sent to Trustweaver is divided by this value to determine the count. **Note:** This value is not editable by the users and is verified by the signature set in trustweaver.sig.1. If either of these values are modified by a user, a Signature Mismatch Error will be posted by the TrustWeaver Request Service. **Example** `unitSizeKB=0`
Archive_Recovery_Folder	Defines the folder in which recovery files from the integrated Archive will be located. **Example** `Archive_Recovery_Folder=&INSTALL_DIR;/data/einvoice`

Property	Description
IntegratedArchive.searchresult.pagesize	Specifies how many search results will be displayed per page in the Integrated Archive. The default is 10. **Example** `IntegratedArchive.searchresult.pagesize=10`
Integratedarchive.reporting.supported countrycodes	Indicates the codes for each country in which validations are reported in the Integrated Archive. Country codes are separated by semi-colons in the list. **Note:** These are the country codes for which the VAT identifier are displayed in the Integrated Archive reports. If the country code is not in this list, then it is considered to be a dummy VAT ID, and is not displayed in the reports. **Note:** **Example** `Integratedarchive.reporting.supportedcountrycodes =AT;BE;BG;CY;CZ;DK;EE;FI;FR;DE;EL;HU;IE;IT;LV;LT;L U;MT;NL;PL;PT;RO;SK;SI;ES;SE;GB`
IntegratedArchive.ui.locale.language	Defines the default language used for the Integrated Archive user interface. The default is en (English). **Example** `IntegratedArchive.ui.locale.language=en`
IntegratedArchive.ui.locale.country	Defines the country code used for the Integrated Archive user interface. The default is us (United States). **Example** `IntegratedArchive.ui.locale.country=us`
IntegratedArchive.ui.decimalformat. pattern	Defines the default decimal format for the Integrated Archive. The default is ######.00 (or 123456.00). **Example** `IntegratedArchive.ui.decimalformat.pattern=######. 00`
IntegratedArchive.ui.dateformat.pattern	Specifies the default date format for the Integrated Archive. The default is yyyy-MM-dd (or 2008-04-01 for April 1, 2008). **Example** `IntegratedArchive.ui.dateformat.pattern=yyyy-MM-dd`
IntegratedArchive.ui.datetimeformat. pattern	Specifies the default date/time format for the Integrated Archive. The default is yyyy-MM-dd hh:mm a (or 2008-04-01 02:04 p for April 1, 2008, 2:04 pm). **Example** `IntegratedArchive.ui.datetimeformat.pattern= yyyy-MM-dd hh:mm a`
timestamping.countrycodes	Specifies the code for each country for which timestamping is automatically performed. The default is IT (Italy). **Example** `timestamping.countrycodes ='IT'`

Property	Description
local.identity	Enables you to specify a default identity to use for creating agreements. **Example** `local.identity=IDENTITY123`
emailThrottle	Enables you to control the number of e-mail messages sent to a partner or Sterling e-Invoice Gateway administrator. Valid values are true (default - control the number of e-mail messages sent) and False. If a problem is causing repeated failures in the e-Invoice business processes (for example, if there is a missing or bad canonical map or lack of connectivity to the integrated archive), setting this parameter to **True** ensures that the Sterling e-Invoice Gateway administrator/operator or partner notification contact is not “spammed” with repeated e-mails. Unchecked, the rate of notification e-mails could overwhelm a customer's e-mail system and personal e-mail account. The **emailThrottle**, **emailLimit**, and **emailTimePeriod** parameters work together to check the rate of notification e-mails. **Example** `emailThrottle=true`
emailLimit	Defines the number email notifications to be sent within the specified time period. If any e-mail notifications were not sent during the specified time period because it would exceed the set limit, a count of how many were skipped will be sent in the next e-mail. **Note:** If a problem is causing repeated failures in the e-Invoice business processes (for example, if there is a missing or bad canonical map or lack of connectivity to the integrated archive), setting this parameter to a low number so that the Sterling e-Invoice Gateway administrator/operator or partner notification contact is not “spammed” with repeated e-mails. Unchecked, the rate of notification e-mails could overwhelm a customer's e-mail system and personal e-mail account. The **emailThrottle**, **emailLimit**, and **emailTimePeriod** parameters work together to check the rate of notification e-mails. **Example** `emailLimit=10`

Property	Description
emailTimePeriod	Defines time period (in minutes) in which the number of notification e-mails sent is to be controlled. The e-mail notification limit will be reset after this time period. The time mentioned here is in minutes. **Note:** If a problem is causing repeated failures in the e-Invoice business processes (for example, if there is a missing or bad canonical map or lack of connectivity to the integrated archive), setting this parameter to a low number so that the Sterling e-Invoice Gateway administrator/operator or partner notification contact is not “spammed” with repeated e-mails. Unchecked, the rate of notification e-mails could overwhelm a customer's e-mail system and personal e-mail account. The **emailThrottle**, **emailLimit**, and **emailTimePeriod** parameters work together to check the rate of notification e-mails. **Example** `emailTimePeriod=1`
IntegratedArchive.searchresult.maxnoofinvoices	Determines the number of invoices that can be returned in a search of the Integrated Archive. **Example** `IntegratedArchive.searchresult.maxnoofinvoices=1500`

messageentryworkstation.properties

The messageentryworkstation.properties file is used to set global configuration parameters for the Message Entry Workstation.

Configuration Settings

The following table describes properties used to configure the messageentryworkstation.properties file in the application:

Property	Description
MSGTYPE-[type_number]	Specifies the display string for the particular message type. **Example** `MSGTYPE-2=SWIFTMX Cash Reporting`
MSGTYPE-[type_number].licenseTag	Specifies the license tag for the particular message type. **Example** `MSGTYPE-2.licenseTag=SWIFTMX`
MSGTYPE-[type_number].versions	Specifies standard versions for the particular message type. Format is standard.versions=comma separated version numbers. **Example** `MSGTYPE-2.versions=1.0,3.0,3.1`
MSGTYPE-[type_number].[stds_version]	Specifies the filename for the message type and standards version for the particular message type. Format is standard.version=file name. **Example** `MSGTYPE-2.1.0=swift_mx_cr_1.0.xml`
enveloping.messageentry	Defines the default message entry envelope used. **Example** `enveloping.messageentry=SWIFTMessageEntryOutbound`
messageentry.SWIFT.fireEvent	If set to True, sends out an e-mail notification when an event occurs. An event occurs when the message status changes from the following: ◆ Draft to ReadyToSend ◆ ReadyToSend to Rejected If set to False, this parameter turns off event notification. **Example** `messageentry.SWIFT.fireEvent=true`
messageentry.SWIFT.READY_TO_SEND.default	The user to which an e-mail should be sent when a message is ready to be sent. **Example** `messageentry.SWIFT.READY_TO_SEND.default=&SI_ADMIN_MAIL_ADDR;`

Property	Description
messageentry.SWIFT.REJECTED.default	The user an mail should be sent when a message is rejected. **Example** `messageentry.SWIFT.REJECTED.default=&SI_ADMIN_MAIL_ADDR;`

restoretablelist.properties

The restoretablelist.properties file is used to set the history used when a table is restored from an archive.

Configuration Settings

The following table describes properties used to configure the restoretablelist.properties file in the application:

Property	Description
FS_OUTBOUND_HISTY	The columns to use when a table containing outbound messages is restored from archive. **Example** `FS_OUTBOUND_HISTY=OBJECT_ID|TIMESTAMP_RECVD|MSG_FORMAT|PARTICIPANT_NUM|TIMESTAMP_SENT|SENT|TIMESTAMP_ACKED|ACKED|MSG_REF|FS_MSG_NUM|MB_MSG_ID|RESENT_COUNT|MSG_TYPE|ADAPTER_NAME|ACKED_MB_MSG_ID|WORKFLOW_ID|DOC_ID|`
FS_INBOUND_HISTY	The columns to use when a table containing inbound messages is restored from archive. **Example** `FS_INBOUND_HISTY=OBJECT_ID|TIMESTAMP_RECVD|MSG_TYPE|PARTICIPANT_NUM|FS_MSG_NUM|MB_MSG_ID|ADAPTER_NAME|TIMESTAMP_ACKED|ACKED|ACKED_MB_MSG_ID|WORKFLOW_ID|DOC_ID|`
DATA_TABLE	???? **Example**
TRANS_DATA	???? **Example**
DOCUMENT	???? **Example**
DOCUMENT_EXTENSION	???? **Example**
WORKFLOW_CONTEXT	???? **Example**
WF_INST_S	???? **Example**

Property	Description
SI_VERSION	???? **Example**
CORRELATION_SET	???? **Example**
EDIINTDOC	???? **Example**
MSGMDNDUP	???? **Example**
MSGMDNCORRELATION	???? **Example**
WEBX_MINED_DATA	???? **Example**

restoretablemap.properties

The restoretablelist.properties file is used to specify the map used when a table is restored from an archive.

Configuration Settings

The following table describes properties used to configure the restoretablemap.properties file in the application:

Property	Description
FS_OUTBOUND_HISTY	Defines the map used when a table containing outbound messages is restored from archive. **Example** `FS_OUTBOUND_HISTY=FS_OUTBND_RESTORE`
FS_INBOUND_HISTY	Defines the map used when a table containing inbound messages is restored from archive. **Example** `FS_INBOUND_HISTY=FS_INBND_RESTORE`

translator_swift.properties

The translator_swift.properties file is used to set default global configuration parameters for the SWIFT standard.

The following caveats apply to this properties file:

- The properties have the following format:

  `swiftField.<fieldTag>.<subfield>.<component>.<mt>[ext]`

- If a field tag has different validation requirements within different sequences of the same message type the following sequence is appended:

  `swiftField.<fieldTag>.<subfield>.<component>.<mt>[ext].<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96 where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.
- 16R/16S code word validations are handled internally by the translator through keyfield matching. Therefore, the entries for these field tags are currently not used.

Note: The properties for 16R and 16S are not enough for performing validation. For example, MT307 requires GENL for 16R, Sequence A and LINK for 16R, Subsequence A1. However, these properties do restrict the code words to the specified lists.

To complete the validation the map editor must use keyfield matching and match against a constant (for example, GENL or LINK). If data is not matched, the translator notes a T13 error code.

- A slash (/) or double slash (//) is always used as the start of a new subfield. Therefore, component and subfield numbers do not exactly match the SWIFT documentation (for example, 41A) when an expanded special function contains subfields or more than one component.
- Field 22, subfield 2, component 1 is matched with a related field that is dependent on the message type. This validation is handled through the <SB-LC> special function.

Code Word Validations for ISO 15022 Messages

The following caveats apply to this properties file for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist).
- The exceptions list has the following format:

  `swiftField.Exceptions.<mt>[ext]`

- The code word lists are displayed by position and order within the MT.
- Non-exception entries have the following format:

  `swiftField.<fieldTag>.<qualifier>.<mt>[ext]`

- Exception entries have the following format:

  `swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]`

- Where a subfield greater than two needs to be validated the subfield number is appended:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield
  >
  ```

- Code words are single-quoted and separated by a comma delimiter.
- Code word lists have the following format:

  ```
  '<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>
  ```

- NODSS indicates that the code word validation is only performed if no DSS is present.
- The <errorCode> will always be the corresponding Kxx code of the field tag.

Special Related Field Tag Properties

Special Related Field Tag properties are used for performing validation when the validation is dependent on a related field. The two cases in which related field processing occurs is for <SB-LC> and for validating an <AMOUNT> where the currency is declared in another field tag.

All related fields must be included with their respective Sequence letters.

Configuration Settings

The following table describes properties used to configure the translator_swift_properties file in the application:

Property	Description
swiftField.[fieldTag].[subfield].[component].[mt]	Defines MT-specific properties. **Example** `swiftField.12.1.1.973='971','972','998':T88`
swiftField.[fieldTag].[subfield].[component].[mt]<ext>.[sequence]	Defines MT-specific properties for the specified sequence. **Example** `swiftField.95S.ALTE.D1.SETPRTY.515='ARNU','CCPT','CHTY','CORP','DRLC','FIIN','TXID':NODSS:K95`
swiftField.[fieldTag].[qualifier].[mt]<ext>	Defines MT-specific properties. **Example** `swiftField.22H.GALO.308='GAIN','LOSS':K22`
swiftField.Exceptions.[mt]<ext>	Defines MT-specific properties for the specified exception. **Example** `swiftField.Exceptions.506=22F.COLA,22H.COLA`
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`

Property	Description
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`

Property	Description
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=100,200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=100,102,103,104,107,110,195, 200,201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`

Property	Description
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`

Property	Description
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 0F,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J,92K,93A ,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P,95Q,95R, 95S,95T,95U,95V,97A,97B,97C,98A,98B,98C,98D,99A, 99B`
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56D,57A,57B,57D,58A,58B,58D,82A,82B,82D,83A, 83D,84A,84B,84D,85A,85B,85D,86A,86B,86D,87A,87B, 87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`
fieldTags.specialExceptions.useSpecificSyntax	Defines MT-specific properties for special cases and exceptions that use a specific syntax. **Example** `fieldTags.specialExceptions.useSpecificSyntax=50F`
fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21, 21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95V`
fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.oneOptionMustBePresent=52B,53B,54B,57B, 58B,85B,86B,87B,88B`
fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77T`

Property	Description
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`
fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D, 15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77T`
fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77T`
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77T`
fieldTags.PARTYFLDJ.format [format_number]	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J, 84J,85J,86J,87J,88J`
fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation=12,31X,35B,82S,68B,68C ,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D,58D, 59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70 ,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`

Property	Description
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D, 53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D ,88D`
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Note:** For SUN customers only. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.relatedFields.[number]	Defines special related field tag properties. **Example** `fieldTags.relatedFields.102=B.32B`
fieldTags.relatedFields.Optional.[number]	Defines related fields that do not have to exist. If the field tags do not exist, then a validation error should not be reported at the end of translation when deferred validation entries are processed. **Example** `fieldTags.relatedFields.Optional.206=C.32A`

Property	Description
fieldTags.relatedFields.Choice. [message_type].[sequence].[field_tag]	Defines related fields that are part of a choice. If one of the choice fields exists, then the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag. **Example** `fieldTags.relatedFields.Choice.405.B.32A=B.32A,B.32K`
fieldTags.SB-LC.relatedFields.subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.[field_tag].relatedFields. [message_type].[sequence].[field]	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.600.A.22=A.33G`
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E,93C,98D`

fieldTags.rule.3.100	Defines special rule properties. These are rules that have to be implemented within the translator because they require validations using external code lists.
	Examples
	`fieldTags.rule.3.100=32A,72`
	`fieldTags.rule.5.107.B=52A,57A`
	`fieldTags.rule.101.303.C=39P`
	`fieldTags.rule.257.210=50C`

fieldTags.rule.32.303.D1	Defines special rule 32 properties. These entries are exceptions to the normal C32 processing. **Example** `fieldTags.rule.32.303.D1=32B,57A,57D,57J`
syntax.[fieldtag][.messageType [extension]]	Defines syntax for a SWIFT Field Tag. If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Example** `syntax.103.97=3!a`
syntax.genericCodeWord.[field_tag]	Defines special function-related syntaxes. Example `syntax.genericCodeWord.69C=':'4!c'/'[8c]'/'<DATE4>'/'4!c`
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines a syntax for a non-generic field. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`

Property	Description
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':'"(.*)"`
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/'"(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`

Property	Description
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`

Property	Description
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`

syntax.INSTR	Defines syntax for the special function <INSTR>.
	Example
	`syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|`
	`('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])`
	`|('/'6!c'/'[27x])|('/'7!c'/'[26x])|`
	`('/'8!c'/'[25x])`

syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`

Property	Description
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF'<35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10x'CRLF'))35x['CRLF'<35B/ISITC/2>]`
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`
syntax.FLD50F_GENERIC	Defines generic syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_GENERIC=35x`
syntax.FLD50F_SPECIFIC	Defines specific syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_SPECIFIC=('/'34x)|(4!a'/'30x)`
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`

Property	Description
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`
syntax.ERIFLD72_STRUCTURED_ GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR> ['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_ SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`

Property	Description
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`

Property	Description
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((''/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((''/'2!c2!n'/')|('/X'1c2!n'/'))[44x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_NEGATIVE	Defines the code word Amount is negative. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`

Property	Description
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`

Property	Description
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.10.subfield=3`
codeWords.REJTRETN72	Defines special function code word lists. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`
codeWords.ISITC2.ASCT	Defines special function code word lists. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`
codeWords.PARTYFLDJ	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`

Property	Description
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

codeWords.rule.[rule_number]	Defines special function code word lists.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_2002.properties

The translator_swift.properties file is used to set default global configuration parameters for the 2002 version of the SWIFT standard.

The following caveats apply to this properties file:

- For each MT (message type) the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `swiftField.Exceptions.<mt>[ext]`
- The code word lists are displayed by position and order in the MT.
- Non-exception entries have the following format:

  `swiftField.<fieldTag>.<qualifier>.<mt>[ext]`
- Exception entries have the following format:

  `swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]`
- When a subfield greater than two must be validated, the subfield number is appended as follows:

  `swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>`
  `swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`
- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`
- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.
- All related fields must be included with their respective Sequence letters.
- If the field tags do not exist, a validation error is not reported at the end of translation when deferred validation entries are processed.
- If one of the choice fields exists for related fields that are part of a choice, the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag.

Configuration Settings

The following table describes properties used to configure the translator_swift_2002.properties file in the application:

Property	Description
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`

Property	Description
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`

Property	Description
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=100,200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=100,102,103,104,107,110,195, 200,201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`

Property	Description
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`

Property	Description
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 0F,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J,92K,93A ,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P,95Q,95R, 95S,95T,95U,95V,97A,97B,97C,98A,98B,98C,98D,99A, 99B`
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56D,57A,57B,57D,58A,58B,58D,82A,82B,82D,83A, 83D,84A,84B,84D,85A,85B,85D,86A,86B,86D,87A,87B, 87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`
fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21, 21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95U`
fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.oneOptionMustBePresent=52B,53B,54B,57B, 58B,85B,86B,87B,88B`
fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77S,77T`
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`

Property	Description
fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D, 15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77S, 77T`
fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77S,77T`
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77S,77T`
fieldTags.PARTYFLDJ.format [format_number]	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J, 84J,85J,86J,87J,88J`
fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation=12,31X,35B,82S,68B,68C ,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D,58D, 59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70 ,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`

Property	Description
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D, 53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D ,88D`
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.[field_tag].relatedFields. subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E, 93C,98D`
fieldTags.rule.3.100	Defines Rule 3 properties. **Example** `fieldTags.rule.3.100=32A,72`
syntax.[fieldtag].[message_type_ extension]	Defines syntax for a SWIFT Field Tag. Format: syntax.<fieldTag>[.messageType[extension]]=<syntax> If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Note:** Only MT100 is supported for the 2002 standard. **Example** `syntax.20=16x`

Property	Description
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines syntax for a non-generic field. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':' "(.*)"`
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/' "(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`

Property	Description
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`

Property	Description
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`
syntax.INSTR	Defines syntax for the special function <INSTR>. **Example** `syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])|('/'6!c'/'[27x])|('/'7!c'/'[26x])|('/'8!c'/'[25x])`
syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`

Property	Description
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF' <35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10 x'CRLF'))35x['CRLF'<35B/ISITC/2>]`
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`

Property	Description
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`
syntax.ERIFLD72_STRUCTURED_GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`

Property	Description
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`

Property	Description
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF' (('/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/' 16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT> 15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF' (('/'2!c2!n'/')|('/X'1c2!n'/'))[44x]'CRLF''/MREF/' 16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT> 15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_ NEGATIVE	Defines the code word Amount is negative. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`

Property	Description
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`

Property	Description
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.10.subfield=3`
codeWords.REJTRETN72	Defines special function code word lists. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`
codeWords.ISITC2.ASCT	Defines special function code word lists. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`

Property	Description
codeWords.PARTYFLDJ	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

codeWords.rule.[rule_number]	Defines special function code word lists.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_2005.properties

The translator_swift_2005.properties file is used to set global configuration parameters for SWIFT 2005 standards.

The following caveats apply to this properties file:

- The entries have the following format:

  ```
  swiftField.[fieldTag].[subfield].[component].[message]
  ```

- The properties for 16R and 16S are not enough for performing validation. For example, MT307 requires GENL for 16R, Sequence A and LINK for 16R, Subsequence A1. However, these properties do restrict the code words to the specified lists in the file.
- For 16R and 16 S, to complete the validation, the Map Editor must use keyfield matching and match against a constant (for example, GENL or LINK). If data is not matched then the translator notes a T13 error code.
- Field 22, subfield 2, component 1 is matched with a related field that depends on the message type. This validation is handled through the <SB-LC> special function.
- Some field tags allow any valid currency in addition to a code word validation. The CUR keyword triggers this validation which checks the external Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply to this properties file for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist).
- The exceptions list has the following format:

  ```
  swiftField.Exceptions.<mt>[ext]
  ```

- The code word lists are displayed by position and order within the MT.
- Non-exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext]
  ```

- Exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]
  ```

- Where a subfield greater than two needs to be validated the subfield number is appended:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>
  ```

- Code words are single-quoted and separated by a comma delimiter.
- Code word lists have the following format:

  ```
  '<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>
  ```

- NODSS indicates that the code word validation is only performed if no DSS is present.
- The <errorCode> will always be the corresponding Kxx code of the field tag.
- If the field tags do not exist for related fields, a validation error is not reported at the end of translation when deferred validation entries are processed.

- If one of the choice fields exists for related fields that are part of a choice, the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag.

Configuration Settings

The following table describes properties used to configure the translator_swift_2005.properties file in the application:

Property	Description
swiftField.Exceptions.[message_type]	Defines swiftField properties. **Example** `swiftField.Exceptions.504=22F.STCO`
swiftField.[fieldTag].[qualifier].[message_type]	Defines swiftField properties. **Example** `swiftField.36B.MILT.518='AMOR','FAMT','UNIT':K36`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type]	Defines swiftField properties. **Example** `swiftField.23E.1.1.104.B='AUTH','NAUT','OTHR':T47`
swiftField.[fieldTag].[qualifier].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.12.1.1.920='940','941','942','950':T88`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.93C.UNBA.B2.ACCTINFO.564.3='ELIG', 'NELG':NODSS:K93`
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`

Property	Description
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`

Property	Description
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=100,200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=100,102,103,104,107,110,195, 200,201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`

Property	Description
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 70F,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J,92K, 93A,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P,95Q, 95R,95S,95T,95U,95V,97A,97B,97C,98A,98B,98C,98D, 99A,99B`

Property	Description
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56D,57A,57B,57D,58A,58B,58D,82A,82B,82D,83A, 83D,84A,84B,84D,85A,85B,85D,86A,86B,86D,87A,87B, 87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`
fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21, 21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95U`
fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.oneOptionMustBePresent=52B,53B,54B,57B, 58B,85B,86B,87B,88B`
fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77T`
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`
fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D, 15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77S, 77T`
fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77S,77T`

Property	Description
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77S,77T`
fieldTags.PARTYFLDJ.format [format_number]	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J, 84J,85J,86J,87J,88J`
fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation=12,31X,35B,82S,68B, 68C,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D, 58D,59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70 ,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D, 53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D ,88D`

Property	Description
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Note:** For SUN customers only. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.relatedFields.[number]	Defines special related field tag properties. **Example** `fieldTags.relatedFields.102=B.32B`
fieldTags.relatedFields.Optional.[number]	Defines related fields that do not have to exist. If the field tags do not exist, then a validation error should not be reported at the end of translation when deferred validation entries are processed. **Example** `fieldTags.relatedFields.Optional.206=C.32A`
fieldTags.relatedFields.Choice. [message_type].[sequence].[field_tag]	Defines related fields that are part of a choice. If one of the choice fields exists, then the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag. **Example** `fieldTags.relatedFields.Choice.405.B.32A=B.32A, B.32K`
fieldTags.[field_tag].relatedFields. subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.[field_tag].relatedFields. [message_type].[sequence].[field]	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.600.A.22=A.33G`
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E, 93C,98D`

Property	Description

fieldTags.rule.[rule_number].[number]	Defines special rule properties. These are rules that have to be implemented within the translator because they require validations using external code lists.
	Examples
	`fieldTags.rule.3.100=32A,72`
	`fieldTags.rule.5.107.B=52A,57A`
	`fieldTags.rule.101.303.C=39P`
	`fieldTags.rule.257.210=50C`

fieldTags.rule.32.303.D1	Defines special rule 32 properties. These entries are exceptions to the normal C32 processing. **Example** `fieldTags.rule.32.303.D1=32B,57A,57D,57J`
syntax.[fieldtag][.messageType [extension]]	Defines syntax for a SWIFT Field Tag. If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Example** `syntax.103.97=3!a`
syntax.genericCodeWord.[function]	Defines special function-related syntaxes. Example `syntax.genericCodeWord.69C=':'4!c'/'[8c]'/'<DATE4>'/'4!c`
syntax.[special_function]	Defines special function-related syntaxes. **Example** `syntax.ISITC2.AEXP=<DATE2>`
syntax.[special_function].[function]	Defines special function-related syntaxes. **Example** `syntax.PARTYFLDJ.NAME='/NAME/'34x['CRLF'34x]0-4`
syntax.regexSun.[field_tag]	Defines special function-related syntaxes. **Note:** For SUN customers only. **Example** `syntax.regexSun.77E=9800x`
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines a syntax for a non-generic field in a System message. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':'"(.*)"`

Property	Description
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/' "(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`

Property	Description
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`

Property	Description
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`
syntax.INSTR	Defines syntax for the special function <INSTR>. **Example** `syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])|('/'6!c'/'[27x])|('/'7!c'/'[26x])|('/'8!c'/'[25x])`
syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`

Property	Description
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF'<35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10x'CRLF'))35x['CRLF'<35B/ISITC/2>]`
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`

Property	Description
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`
syntax.ERIFLD72_STRUCTURED_GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`

Property	Description
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'(('/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`

Property	Description
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'(('/'2!c2!n'/')|('/X'1c2!n'/'))[44x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_NEGATIVE	Defines a negative code word amount. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`

Property	Description
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69D.12.subfield=3`

Property	Description
codeWords.REJTRETN72	Defines special function code word lists. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`
codeWords.ISITC2.ASCT	Defines special function code word lists. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`
codeWords.PARTYFLDJ	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

Property	Description

codeWords.rule.[rule_number]	Defines special function code word lists.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_2006.properties

The translator_swift_2006.properties file is used to set global configuration parameters for SWIFT 2006 standards.

The following caveats apply to this properties file:

- The entries have the following format:

  ```
  swiftField.[fieldTag].[subfield].[component].[message]
  ```

- The properties for 16R and 16S are not enough for performing validation. For example, MT307 requires GENL for 16R, Sequence A and LINK for 16R, Subsequence A1. However, these properties do restrict the code words to the specified lists in the file.
- For 16R and 16 S, to complete the validation, the Map Editor must use keyfield matching and match against a constant (for example, GENL or LINK). If data is not matched then the translator notes a T13 error code.
- Field 22, subfield 2, component 1 is matched with a related field that depends on the message type. This validation is handled through the <SB-LC> special function.
- Some field tags allow any valid currency in addition to a code word validation. The CUR keyword triggers this validation which checks the external Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply to this properties file for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist).
- The exceptions list has the following format:

  ```
  swiftField.Exceptions.<mt>[ext]
  ```

- The code word lists are displayed by position and order within the MT.
- Non-exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext]
  ```

- Exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]
  ```

- Where a subfield greater than two needs to be validated the subfield number is appended:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>
  ```

- Code words are single-quoted and separated by a comma delimiter.
- Code word lists have the following format:

  ```
  '<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>
  ```

- NODSS indicates that the code word validation is only performed if no DSS is present.
- The <errorCode> will always be the corresponding Kxx code of the field tag.
- If the field tags do not exist for related fields, a validation error is not reported at the end of translation when deferred validation entries are processed.

- If one of the choice fields exists for related fields that are part of a choice, the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag.

Configuration Settings

The following table describes properties used to configure the translator_swift_2006.properties file in the application:

Property	Description
swiftField.Exceptions.[message_type]	Defines swiftField properties. **Example** `swiftField.Exceptions.504=22F.STCO`
swiftField.[fieldTag].[qualifier].[message_type]	Defines swiftField properties. **Example** `swiftField.36B.MILT.518='AMOR','FAMT','UNIT':K36`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type]	Defines swiftField properties. **Example** `swiftField.23E.1.1.104.B='AUTH','NAUT','OTHR':T47`
swiftField.[fieldTag].[qualifier].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.12.1.1.920='940','941','942','950':T88`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.93C.UNBA.B2.ACCTINFO.564.3='ELIG', 'NELG':NODSS:K93`
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`

Property	Description
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`

Property	Description
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=102,103,104,107,110,195,200, 201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`

Property	Description
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 70F,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J,92K, 93A,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P,95Q, 95R,95S,95T,95U,95V,97A,97B,97C,98A,98B,98C,98D, 99A,99B`

Property	Description
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56D,57A,57B,57D,58A,58B,58D,82A,82B,82D,83A, 83D,84A,84B,84D,85A,85B,85D,86A,86B,86D,87A,87B, 87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`
fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21, 21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95U,95V`
fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.oneOptionMustBePresent=52B,53B,54B,57B, 58B,85B,86B,87B,88B`
fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77T`
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`
fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D, 15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77T`
fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77T`

Property	Description
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77T`
fieldTags.PARTYFLDJ.format [format_number]	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J, 84J,85J,86J,87J,88J`
fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation=12,31X,35B,82S,68B, 68C,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D, 58D,59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70 ,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D, 53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D ,88D`

Property	Description
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Note:** For SUN customers only. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.relatedFields.[number]	Defines special related field tag properties. **Example** `fieldTags.relatedFields.102=B.32B`
fieldTags.relatedFields.Optional.[number]	Defines related fields that do not have to exist. If the field tags do not exist, then a validation error should not be reported at the end of translation when deferred validation entries are processed. **Example** `fieldTags.relatedFields.Optional.206=C.32A`
fieldTags.relatedFields.Choice. [message_type].[sequence].[field_tag]	Defines related fields that are part of a choice. If one of the choice fields exists, then the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag. **Example** `fieldTags.relatedFields.Choice.405.B.32A=B.32A, B.32K`
fieldTags.[field_tag].relatedFields. subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.[field_tag].relatedFields. [message_type].[sequence].[field]	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.600.A.22=A.33G`
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E, 93C,98D`

Property	Description

fieldTags.rule.[rule_number].[number]	Defines special rule properties. These are rules that have to be implemented within the translator because they require validations using external code lists.
	Examples
	`fieldTags.rule.3.100=32A,72`
	`fieldTags.rule.5.107.B=52A,57A`
	`fieldTags.rule.101.303.C=39P`
	`fieldTags.rule.257.210=50C`

fieldTags.rule.32.303.D1	Defines special rule 32 properties. These entries are exceptions to the normal C32 processing. **Example** `fieldTags.rule.32.303.D1=32B,57A,57D,57J`
syntax.[fieldtag][.messageType [extension]]	Defines syntax for a SWIFT Field Tag. If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Example** `syntax.103.97=3!a`
syntax.genericCodeWord.[field_tag]	Defines special function-related syntaxes. Example `syntax.genericCodeWord.69C=':'4!c'/'[8c]'/'<DATE4>'/'4!c`
syntax.[special_function]	Defines special function-related syntaxes. **Example** `syntax.ISITC2.AEXP=<DATE2>`
syntax.[special_function].[function]	Defines special function-related syntaxes. **Example** `syntax.PARTYFLDJ.NAME='/NAME/'34x['CRLF'34x]0-4`
syntax.regexSun.[field_tag]	Defines special function-related syntaxes. **Note:** For SUN customers only. **Example** `syntax.regexSun.77E=9800x`
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines a syntax for a non-generic field. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':'"(.*)"`

Property	Description
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/' "(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`

Property	Description
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`

Property	Description
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`
syntax.INSTR	Defines syntax for the special function <INSTR>. **Example** `syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])|('/'6!c'/'[27x])|('/'7!c'/'[26x])|('/'8!c'/'[25x])`
syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`

Property	Description
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF'<35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10x'CRLF'))35x['CRLF'<35B/ISITC/2>]`
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`

Property	Description
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`
syntax.ERIFLD72_STRUCTURED_GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`

Property	Description
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'(('/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`

Property	Description
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'(('/'2!c2!n'/')|('/X'1c2!n'/'))[44x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_NEGATIVE	Defines a negative code word amount. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`

Property	Description
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69D.12.subfield=3`

Property	Description
codeWords.REJTRETN72	Defines special function code word lists. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`
codeWords.ISITC2.ASCT	Defines special function code word lists. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`
codeWords.PARTYFLDJ	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

Property	Description

codeWords.rule.[rule_number]	Defines special function code word lists.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_2007.properties

The translator_swift_2007.properties file is used to set global configuration parameters for SWIFT 2007 standards.

The following caveats apply to this properties file:

- The entries have the following format:

  ```
  swiftField.[fieldTag].[subfield].[component].[message]
  ```

- The properties for 16R and 16S are not enough for performing validation. For example, MT307 requires GENL for 16R, Sequence A and LINK for 16R, Subsequence A1. However, these properties do restrict the code words to the specified lists in the file.
- For 16R and 16 S, to complete the validation, the Map Editor must use keyfield matching and match against a constant (for example, GENL or LINK). If data is not matched then the translator notes a T13 error code.
- Field 22, subfield 2, component 1 is matched with a related field that depends on the message type. This validation is handled through the <SB-LC> special function.
- Some field tags allow any valid currency in addition to a code word validation. The CUR keyword triggers this validation which checks the external Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply to this properties file for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist).
- The exceptions list has the following format:

  ```
  swiftField.Exceptions.<mt>[ext]
  ```

- The code word lists are displayed by position and order within the MT.
- Non-exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext]
  ```

- Exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]
  ```

- Where a subfield greater than two needs to be validated the subfield number is appended:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>
  ```

- Code words are single-quoted and separated by a comma delimiter.
- Code word lists have the following format:

  ```
  '<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>
  ```

- NODSS indicates that the code word validation is only performed if no DSS is present.
- The <errorCode> will always be the corresponding Kxx code of the field tag.
- If the field tags do not exist for related fields, a validation error is not reported at the end of translation when deferred validation entries are processed.

- If one of the choice fields exists for related fields that are part of a choice, the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag.

Configuration Settings

The following table describes properties used to configure the translator_swift_2007.properties file in the application:

Property	Description
swiftField.Exceptions.[message_type]	Defines swiftField properties. **Example** `swiftField.Exceptions.504=22F.STCO`
swiftField.[fieldTag].[qualifier].[message_type]	Defines swiftField properties. **Example** `swiftField.36B.MILT.500='AMOR','FAMT','UNIT':K36`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type]	Defines swiftField properties. **Example** `swiftField.23E.1.1.104.B='AUTH','NAUT','OTHR':T47`
swiftField.[fieldTag].[qualifier].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.12.1.1.920='940','941','942','950':T88`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.93C.UNBA.B2.ACCTINFO.564.3='ELIG', 'NELG':NODSS:K93`
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`

Property	Description
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`

Property	Description
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=102,103,104,107,110,195,200, 201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`

Property	Description
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 70F,70G,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J, 92K,93A,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P, 95Q,95R,95S,95T,95U,95V,97A,97B,97C,98A,98B,98C, 98D,99A,99B`

Property	Description
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56B,56D,56F,57A,57B,57D,58A,58B,58D,82A,82B, 82D,83A,83D,84A,84B,84D,85A,85B,85D,86A,86B,86D, 87A,87B,87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,50F,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`
fieldTags.specialExceptions. useSpecificSyntax	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions.useSpecificSyntax=50F`
fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21, 21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95U,95V`
fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.oneOptionMustBePresent=52B,53B,54B,56B, 57B,58B,85B,86B,87B,88B`
fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77T`
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`
fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D, 15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77T`

Property	Description
fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77T`
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77T`
fieldTags.PARTYFLDJ.format [format_number]	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J, 84J,85J,86J,87J,88J`
fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation=12,31X,35B,82S,68B, 68C,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D, 58D,59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70 ,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`

Property	Description
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D, 53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D ,88D`
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Note:** For SUN customers only. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.relatedFields.[number]	Defines special related field tag properties. **Example** `fieldTags.relatedFields.102=B.32B`
fieldTags.relatedFields.Optional.[number]	Defines related fields that do not have to exist. If the field tags do not exist, then a validation error should not be reported at the end of translation when deferred validation entries are processed. **Example** `fieldTags.relatedFields.Optional.206=C.32A`
fieldTags.relatedFields.Choice. [message_type].[sequence].[field_tag]	Defines related fields that are part of a choice. If one of the choice fields exists, then the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag. **Example** `fieldTags.relatedFields.Choice.405.B.32A=B.32A, B.32K`
fieldTags.[field_tag].subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.[field_tag].relatedFields. [message_type].[sequence].[field]	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.600.A.22=A.33G`

Property	Description
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E, 93C,98D`

fieldTags.rule.[rule_number].[number]	Defines special rule properties. These are rules that have to be implemented within the translator because they require validations using external code lists.
	Examples
	`fieldTags.rule.3.100=32A,72`
	`fieldTags.rule.5.107.B=52A,57A`
	`fieldTags.rule.101.303.C=39P`
	`fieldTags.rule.257.210=50C`

fieldTags.rule.32.303.D1	Defines special rule 32 properties. These entries are exceptions to the normal C32 processing. **Example** `fieldTags.rule.32.303.D1=32B,57A,57D,57J`
syntax.[fieldtag][.messageType [extension]]	Defines syntax for a SWIFT Field Tag. If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Example** `syntax.103.97=3!a`
syntax.genericCodeWord.[field_tag]	Defines special function-related syntaxes. Example `syntax.genericCodeWord.69C=':'4!c'/'[8c]'/'<DATE4> '/'4!c`
syntax.[special_function]	Defines special function-related syntaxes. **Example** `syntax.ISITC2.AEXP=<DATE2>`
syntax.[special_function].[format]	Defines special function-related syntaxes. **Example** `syntax.PARTYFLDJ.NAME='/NAME/'34x['CRLF'34x]0-4`
syntax.regexSun.[field_tag]	Defines special function-related syntaxes. **Note:** For SUN customers only. **Example** `syntax.regexSun.77E=9800x`
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines a syntax for a non-generic field. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`

Property	Description
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':' "(.*)"`
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/' "(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`

Property	Description
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`

Property	Description
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`

syntax.INSTR	Defines syntax for the special function <INSTR>.
	Example
	`syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|`
	`('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])`
	`|('/'6!c'/'[27x])|('/'7!c'/'[26x])|`
	`('/'8!c'/'[25x])`

syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`

Property	Description
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF'<35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10x'CRLF'))35x['CRLF'<35B/ISITC/2>]`
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`
syntax.FLD50F_GENERIC	Defines generic syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_GENERIC=35x`
syntax.FLD50F_SPECIFIC	Defines specific syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_SPECIFIC=('/'34x)|(4!a'/'30x)`
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`

Property	Description
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`
syntax.ERIFLD72_STRUCTURED_GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR> ['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`

Property	Description
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`

Property	Description
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((''/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((''/'2!c2!n'/')|('/X'1c2!n'/'))[44x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_NEGATIVE	Defines a negative code word amount. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`

Property	Description
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`

Property	Description
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69D.12.subfield=3`
codeWords.REJTRETN72	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`

Property	Description
codeWords.ISITC2.ASCT	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`
codeWords.PARTYFLDJ	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

codeWords.rule.[rule_number]	Defines special function code word lists.
	Note: These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_2008.properties

The translator_swift_2008.properties file is used to set global configuration parameters for SWIFT 2008 standards.

The following caveats apply to this properties file:

- The entries have the following format:

  ```
  swiftField.[fieldTag].[subfield].[component].[message]
  ```

- The properties for 16R and 16S are not enough for performing validation. For example, MT307 requires GENL for 16R, Sequence A and LINK for 16R, Subsequence A1. However, these properties do restrict the code words to the specified lists in the file.
- For 16R and 16 S, to complete the validation, the Map Editor must use keyfield matching and match against a constant (for example, GENL or LINK). If data is not matched then the translator notes a T13 error code.
- Field 22, subfield 2, component 1 is matched with a related field that depends on the message type. This validation is handled through the <SB-LC> special function.
- Some field tags allow any valid currency in addition to a code word validation. The CUR keyword triggers this validation which checks the external Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply to this properties file for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist).
- The exceptions list has the following format:

  ```
  swiftField.Exceptions.<mt>[ext]
  ```

- The code word lists are displayed by position and order within the MT.
- Non-exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext]
  ```

- Exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]
  ```

- Where a subfield greater than two needs to be validated the subfield number is appended:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>
  ```

- Code words are single-quoted and separated by a comma delimiter.
- Code word lists have the following format:

  ```
  '<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>
  ```

- NODSS indicates that the code word validation is only performed if no DSS is present.
- The <errorCode> will always be the corresponding Kxx code of the field tag.
- If the field tags do not exist for related fields, a validation error is not reported at the end of translation when deferred validation entries are processed.

- If one of the choice fields exists for related fields that are part of a choice, the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag.

Configuration Settings

The following table describes properties used to configure the translator_swift_2008.properties file in the application:

Property	Description
swiftField.Exceptions.[message_type]	Defines swiftField properties. **Example** `swiftField.Exceptions.504=22F.STCO`
swiftField.[fieldTag].[qualifier].[message_type]	Defines swiftField properties. **Example** `swiftField.36B.MILT.500='AMOR','FAMT','UNIT':K36`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type]	Defines swiftField properties. **Example** `swiftField.23E.1.1.104.B='AUTH','NAUT','OTHR':T47`
swiftField.[fieldTag].[qualifier].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.12.1.1.920='940','941','942','950':T88`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.93C.UNBA.B2.ACCTINFO.564.3='ELIG', 'NELG':NODSS:K93`
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`

Property	Description
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`

Property	Description
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=102,103,104,107,110,195,200, 201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`

Property	Description
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 70F,70G,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J, 92K,93A,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P, 95Q,95R,95S,95T,95U,95V,97A,97B,97C,98A,98B,98C, 98D,99A,99B`

Property	Description
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56B,56D,56F,57A,57B,57D,58A,58B,58D,82A,82B, 82D,83A,83D,84A,84B,84D,85A,85B,85D,86A,86B,86D, 87A,87B,87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,50F,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`
fieldTags.specialExceptions. useSpecificSyntax	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions.useSpecificSyntax=50F`
fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21, 21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95U,95V`
fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.oneOptionMustBePresent=52B,53B,54B,56B, 57B,58B,85B,86B,87B,88B`
fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77T`
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`
fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D, 15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77T`

Property	Description
fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77T`
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77T`
fieldTags.PARTYFLDJ.format [format_number]	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J, 84J,85J,86J,87J,88J`
fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation=12,31X,35B,82S,68B, 68C,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D, 58D,59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70 ,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`

Property	Description
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D, 53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D ,88D`
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Note:** For SUN customers only. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.relatedFields.[number]	Defines special related field tag properties. **Example** `fieldTags.relatedFields.102=B.32B`
fieldTags.relatedFields.Optional.[number]	Defines related fields that do not have to exist. If the field tags do not exist, then a validation error should not be reported at the end of translation when deferred validation entries are processed. **Example** `fieldTags.relatedFields.Optional.206=C.32A`
fieldTags.relatedFields.Choice. [message_type].[sequence].[field_tag]	Defines related fields that are part of a choice. If one of the choice fields exists, then the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag. **Example** `fieldTags.relatedFields.Choice.405.B.32A=B.32A, B.32K`
fieldTags.[field_tag].subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.[field_tag].relatedFields. [message_type].[sequence].[field]	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.600.A.22=A.33G`

Property	Description
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E, 93C,98D`

fieldTags.rule.[rule_number].[number]	Defines special rule properties. These are rules that have to be implemented within the translator because they require validations using external code lists.
	Examples
	`fieldTags.rule.3.100=32A,72`
	`fieldTags.rule.5.107.B=52A,57A`
	`fieldTags.rule.101.303.C=39P`
	`fieldTags.rule.257.210=50C`

fieldTags.rule.32.303.D1	Defines special rule 32 properties. These entries are exceptions to the normal C32 processing. **Example** `fieldTags.rule.32.303.D1=32B,57A,57D,57J`
syntax.[fieldtag][.messageType [extension]]	Defines syntax for a SWIFT Field Tag. If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Example** `syntax.103.97=3!a`
syntax.genericCodeWord.[field_tag]	Defines special function-related syntaxes. Example `syntax.genericCodeWord.69C=':'4!c'/'[8c]'/'<DATE4> '/'4!c`
syntax.[special_function]	Defines special function-related syntaxes. **Example** `syntax.ISITC2.AEXP=<DATE2>`
syntax.[special_function].[format]	Defines special function-related syntaxes. **Example** `syntax.PARTYFLDJ.NAME='/NAME/'34x['CRLF'34x]0-4`
syntax.regexSun.[field_tag]	Defines special function-related syntaxes. **Note:** For SUN customers only. **Example** `syntax.regexSun.77E=9800x`
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines a syntax for a non-generic field. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`

Property	Description
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':' "(.*)"`
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/' "(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`

Property	Description
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`

Property	Description
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`
syntax.INSTR	Defines syntax for the special function <INSTR>. **Example** `syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])|('/'6!c'/'[27x])|('/'7!c'/'[26x])|('/'8!c'/'[25x])`
syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`

Property	Description
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF'<35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10x'CRLF'))35x['CRLF'<35B/ISITC/2>]`
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`
syntax.FLD50F_GENERIC	Defines generic syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_GENERIC=35x`
syntax.FLD50F_SPECIFIC	Defines specific syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_SPECIFIC=('/'34x)|(4!a'/'30x)`
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`

Property	Description
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`
syntax.ERIFLD72_STRUCTURED_GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR> ['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`

Property	Description
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`

Property	Description
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((''/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((''/'2!c2!n'/')|('/X'1c2!n'/'))[44x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_NEGATIVE	Defines a negative code word amount. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`

Property	Description
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`

Property	Description
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69D.12.subfield=3`
codeWords.REJTRETN72	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`

Property	Description
codeWords.ISITC2.ASCT	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`
codeWords.PARTYFLDJ	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

codeWords.rule.[rule_number]	Defines special function code word lists.
	Note: These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_2009.properties

The translator_swift_2009.properties file is used to set global configuration parameters for SWIFT 2009 standards.

The following caveats apply to this properties file:

- The entries have the following format:

  ```
  swiftField.[fieldTag].[subfield].[component].[message]
  ```

- The properties for 16R and 16S are not enough for performing validation. For example, MT307 requires GENL for 16R, Sequence A and LINK for 16R, Subsequence A1. However, these properties do restrict the code words to the specified lists in the file.
- For 16R and 16 S, to complete the validation, the Map Editor must use keyfield matching and match against a constant (for example, GENL or LINK). If data is not matched then the translator notes a T13 error code.
- Field 22, subfield 2, component 1 is matched with a related field that depends on the message type. This validation is handled through the <SB-LC> special function.
- Some field tags allow any valid currency in addition to a code word validation. The CUR keyword triggers this validation which checks the external Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply to this properties file for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist).
- The exceptions list has the following format:

  ```
  swiftField.Exceptions.<mt>[ext]
  ```

- The code word lists are displayed by position and order within the MT.
- Non-exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext]
  ```

- Exception entries have the following format:

  ```
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext]
  ```

- Where a subfield greater than two needs to be validated the subfield number is appended:

  ```
  swiftField.<fieldTag>.<qualifier>.<mt>[ext].<subfield>
  swiftField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>
  ```

- Code words are single-quoted and separated by a comma delimiter.
- Code word lists have the following format:

  ```
  '<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>
  ```

- NODSS indicates that the code word validation is only performed if no DSS is present.
- The <errorCode> will always be the corresponding Kxx code of the field tag.
- If the field tags do not exist for related fields, a validation error is not reported at the end of translation when deferred validation entries are processed.

- If one of the choice fields exists for related fields that are part of a choice, the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag.

Configuration Settings

Note: Please refer to the specific sections of the SWIFT documentation (parts II and III) to view the details of how field tag and code word validation is performed for all field tags.

The following table describes properties used to configure the translator_swift_2009.properties file in the application:

Property	Description
swiftField.Exceptions.[message_type]	Defines swiftField properties. **Example** `swiftField.Exceptions.504=22F.STCO`
swiftField.[fieldTag].[qualifier].[message_type]	Defines swiftField properties. **Example** `swiftField.36B.MILT.500='AMOR','FAMT','UNIT':K36`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type]	Defines swiftField properties. **Example** `swiftField.23E.1.1.104.B='AUTH','NAUT','OTHR':T47`
swiftField.[fieldTag].[qualifier].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.12.1.1.920='940','941','942','950':T88`
swiftField.[fieldTag].[qualifier].[sequence].[isoBlock].[message_type].[subfield]	Defines swiftField properties. **Example** `swiftField.93C.UNBA.B2.ACCTINFO.564.3='ELIG', 'NELG':NODSS:K93`
mt.smallInputMaximum	Defines MT-specific properties. **Example** `mt.smallInputMaximum=2000`
mt.smallOutputMaximum	Defines MT-specific properties. **Example** `mt.smallOutputMaximum=2600`
mt.largeInputMaximum	Defines MT-specific properties. **Example** `mt.largeInputMaximum=10000`
mt.largeOutputMaximum	Defines MT-specific properties. **Example** `mt.largeOutputMaximum=10600`
mt.largeInput	Defines MT-specific properties. **Example** `mt.largeInput=101,102,103,104,106,107,121,206,207, 256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`

Property	Description
mt.largeOutput	Defines MT-specific properties. **Example** `mt.largeOutput=101,102,103,104,106,107,121,206,207 ,256,300,303,304,306,307,308,320,321,330,340,341, 350,360,361,364,365,380,381,405,416,500,501,502, 503,504,505,506,507,508,509,510,513,514,515,517, 518,519,524,527,528,529,535,536,537,538,540,541, 542,543,544,545,546,547,548,549,558,564,565,566, 567,568,569,574,575,576,577,578,584,586,587,588, 589,700,701,710,711,720,721,760,767,98,198,298,398 ,498,598,698,798,898,998,9999`
mt.rule.32	Defines MT-specific properties. **Example** `mt.rule.32=300,303,304,306,320,330,340,350,360,361 ,362,364,365,405,582,600,643,645,813`
mtFieldTag.F72.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F72.other=192,292,392,492,592,692,792, 892,992,395,495,595,695,795,895,995,196,296,396, 496,596,696,796,896,996`
mtFieldTag.F79.other	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.F79.other=960,964,965,966,967,192,292, 392,492,592,692,792,892,992,395,495,595,695,795, 895,995,196,296,396,496,596,696,796,896,996,399, 499,599,699,799,899,999`
mtFieldTag.ERI.F72.narrative	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F72.narrative=110,400,410,412,420, 422,430,450,456,516,526,559,577,581,600,601,604, 605,606,607,609,643,644,645,646,700,705,707,710, 720,730,732,734,740,742,747,750,752,754,756,760, 767,768,769,800,802,812,813,820,821,822,823,824, 900,910,935,90,190,290,390,490,590,690,790,890,990 ,91,191,291,391,491,591,691,791,891,991`
mtFieldTag.FLD72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72=200,201,202,203,204,205,206, 300,303,304,305,306,320,330,340,341,350,360,361, 362,364,365,405`
mtFieldTag.FLD72._STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._STP=102,103`
mtFieldTag.FLD72._not_STP	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.FLD72._not_STP=102,103`

Property	Description
mtFieldTag.REJTRETN72	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN72=102,103,104,107,110,195,200, 201,202,203,204,205,207,295`
mtFieldTag.ERI.F77A	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F77A=416,455,456,747,754,810`
mtFieldTag.ERI.F79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.ERI.F79=582,649,705,707,810,986`
mtFieldTag.REJTRETN79	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.REJTRETN79=195,199,295,299`
mtFieldTag.SB-LC.DATE	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.SB-LC.DATE=360,361,362,364,365`
mtFieldTag.rule.89	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89=600,601,604,605,606,607,608,609`
mtFieldTag.rule.89.32F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.32F=600,601,604,605,606,607`
mtFieldTag.rule.89.60F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60F=608`
mtFieldTag.rule.89.60M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.60M=608`
mtFieldTag.rule.89.61	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.61=608`
mtFieldTag.rule.89.62F	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62F=608`
mtFieldTag.rule.89.62M	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.62M=608`
mtFieldTag.rule.89.64	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.64=608`

Property	Description
mtFieldTag.rule.89.65	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.65=608`
mtFieldTag.rule.89.68B.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68B.3.4=609`
mtFieldTag.rule.89.68C.3.4	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.rule.89.68C.3.4=609`
mtFieldTag.specialValidation.T75	Defines MT-specific properties for field tag validation. **Example** `mtFieldTag.specialValidation.T75=405,192,292,392, 492,592,692,792,892,992,195,295,395,495,595,695, 795,895,995,196,296,396,496,596,696,796,896,996`
fieldTags.loopRepeat	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.loopRepeat=15A,15B,15C,15D,15E,15F,15G, 15H,15I,15J,15K,15L,15M,15N`
fieldTags.genericFields	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.genericFields=11A,12A,12B,12C,13A,13B, 13J,13K,17B,19A,19B,20C,20D,22F,22H,24B,25D,36B, 36C,36E,69A,69B,69C,69D,69E,69F,69J,70C,70D,70E, 70F,70G,90A,90B,90E,92A,92B,92C,92D,92E,92F,92J, 92K,93A,93B,93C,93D,94B,94C,94D,94F,94G,95C,95P, 95Q,95R,95S,95T,95U,95V,97A,97B,97C,98A,98B,98C, 98D,99A,99B`
fieldTags.checkCRLFUsage	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFUsage=35B,42A,42D,50D,51A,51D, 52A,52B,52D,53A,53B,53D,54A,54B,54D,55A,55B,55D, 56A,56B,56D,56F,57A,57B,57D,58A,58B,58D,82A,82B, 82D,83A,83D,84A,84B,84D,85A,85B,85D,86A,86B,86D, 87A,87B,87D,88A,88B,88D`
fieldTags.specialExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions=12,12E,12F,14A,14D,14G ,14J,16R,16S,17A,17B,17F,17G,17N,17O,17R,17T,17U, 17V,22,22A,22B,22D,22E,22G,22J,22K,23,23A,23B,23C, 23D,23E,23F,23G,24D,26C,26F,28E,31X,32F,32K,33B, 33G,33K,33S,33T,35A,35H,35N,35S,35U,37K,38B,38E, 38G,38H,38J,39B,39P,40A,40B,40C,40E,40F,41A,41D, 49,50F,60F,61,62F,62M,64,65,68B,68C,71A,77H,94A`

Property	Description
fieldTags.specialExceptions. useSpecificSyntax	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialExceptions.useSpecificSyntax=50F`

fieldTags.noSlashesAllowed	Defines MT-specific properties for special cases and exceptions.
	Example
	`fieldTags.noSlashesAllowed=12A,13B,20,20C,20D,21,`
	`21A,21F,21G,21P,21R,94B,95Q,95R,95S,95T,95U,95V`

fieldTags.oneOptionMustBePresent	Defines MT-specific properties for special cases and exceptions.
	Example
	`fieldTags.oneOptionMustBePresent=52B,53B,54B,56B,`
	`57B,58B,85B,86B,87B,88B`

fieldTags.checkCRLFPairs	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.checkCRLFPairs=77E`
fieldTags.commonMTExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTExceptions=77F,77G,77T`
fieldTags.commonMTGenericExceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.commonMTGenericExceptions=17B`
fieldTags.MT574Exceptions	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.MT574Exceptions=22F`

fieldTags.CRLFStartsLineAllowed	Defines MT-specific properties for special cases and exceptions.
	Example
	`fieldTags.CRLFStartsLineAllowed=15A,15B,15C,15D,`
	`15E,15F,15G,15H,15I,15J,15K,15L,15M,15N,77E,77T`

fieldTags.allBlanksAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.allBlanksAllowed=77E,77T`
fieldTags.colonStartsSecondLineAllowed	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.colonStartsSecondLineAllowed=77E,77T`

fieldTags.PARTYFLDJ.format
[format_number]
	Defines MT-specific properties for special cases and exceptions.
	Example
	`fieldTags.PARTYFLDJ.format1=53J,56J,57J,58J,82J,`
	`84J,85J,86J,87J,88J`

fieldTags.specialValidation	Defines MT-specific properties for special cases and exceptions.
	Example
	`fieldTags.specialValidation=12,31X,35B,82S,68B,`
	`68C,38B,50H,42D,50D,50K,52D,53D,54D,55D,56D,57D,`
	`58D,59,82D,83D,84D,85D,86D,87D,88D`

Property	Description
fieldTags.specialValidation.Errors	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.Errors=T12,T18,T62,T70,T74,T75,T76,T77`
fieldTags.specialValidation.T12	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T12=35B`
fieldTags.specialValidation.T18	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T18=12`
fieldTags.specialValidation.T62	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T62=31X`
fieldTags.specialValidation.T70	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T70=82S`
fieldTags.specialValidation.T74	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T74=68B,68C`
fieldTags.specialValidation.T75	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T75=38B`
fieldTags.specialValidation.T76	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T76=50H`
fieldTags.specialValidation.T77	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.specialValidation.T77=42D,50D,50K,52D,53D,54D,55D,56D,57D,58D,59,82D,83D,84D,85D,86D,87D,88D`
fieldTags.regexSun	Defines MT-specific properties for special cases and exceptions. **Note:** For SUN customers only. **Example** `fieldTags.regexSun=77C,77E`
fieldTags.BICPlus.102_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_not_STP=52A,52C,57A,57C`
fieldTags.BICPlus.102_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.102_STP=52A,57A`

Property	Description
fieldTags.BICPlus.103_not_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_not_STP=52A,52D,56A,56C,56D, 57A,57C,57D`
fieldTags.BICPlus.103_STP	Defines MT-specific properties for special cases and exceptions. **Example** `fieldTags.BICPlus.103_STP=52A,56A,57A`
fieldTags.relatedFields.[number]	Defines special related field tag properties. **Example** `fieldTags.relatedFields.102=B.32B`
fieldTags.relatedFields.Optional.[number]	Defines related fields that do not have to exist. If the field tags do not exist, then a validation error should not be reported at the end of translation when deferred validation entries are processed. **Example** `fieldTags.relatedFields.Optional.206=C.32A`
fieldTags.relatedFields.Choice. [message_type].[sequence].[field_tag]	Defines related fields that are part of a choice. If one of the choice fields exists, then the other field tags do not have to exist. Each message type lists the choice options for a particular sequence and field tag. **Example** `fieldTags.relatedFields.Choice.405.B.32A=B.32A, B.32K`
fieldTags.[field_tag].subfield2	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.subfield2=33G,32B`
fieldTags.[field_tag].relatedFields. [message_type].[sequence].[field]	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.SB-LC.relatedFields.600.A.22=A.33G`
fieldTags.genericCodeWordExceptions	Defines related fields for SB-LC and AMOUNT special functions. **Example** `fieldTags.genericCodeWordExceptions=69C,69D,92E, 93C,98D`

fieldTags.rule.[rule_number].[number]	Defines special rule properties. These are rules that have to be implemented within the translator because they require validations using external code lists.
	Examples
	`fieldTags.rule.3.100=32A,72`
	`fieldTags.rule.5.107.B=52A,57A`
	`fieldTags.rule.101.303.C=39P`
	`fieldTags.rule.257.210=50C`

fieldTags.rule.32.303.D1	Defines special rule 32 properties. These entries are exceptions to the normal C32 processing. **Example** `fieldTags.rule.32.303.D1=32B,57A,57D,57J`

Property	Description
syntax.[fieldtag][.messageType [extension]]	Defines syntax for a SWIFT Field Tag. If a field tag syntax does not have a message-specific syntax, the message type (and extension) are not appended to the syntax entry. **Example** `syntax.103.97=3!a`
syntax.genericCodeWord.[field_tag]	Defines special function-related syntaxes. Example `syntax.genericCodeWord.69C=':'4!c'/'[8c]'/'<DATE4>'/'4!c`
syntax.[special_function]	Defines special function-related syntaxes. **Example** `syntax.ISITC2.AEXP=<DATE2>`
syntax.[special_function].[format]	Defines special function-related syntaxes. **Example** `syntax.PARTYFLDJ.NAME='/NAME/'34x['CRLF'34x]0-4`
syntax.regexSun.[field_tag]	Defines special function-related syntaxes. **Note:** For SUN customers only. **Example** `syntax.regexSun.77E=9800x`
syntax.BICPlus	Set this property to "YES" if BIC Plus validation should be performed. **Example** `syntax.BICPlus=YES`
syntax.NON_GENERIC_FIELD	Defines a syntax for a non-generic field. **Example** `syntax.NON_GENERIC_FIELD=':'2!n[1a]':'"(.*)"`
syntax.NON_GENERIC_SYSTEM_FIELD	Defines syntax for a non-generic field in a System message (for example. 096, 097). **Example** `syntax.NON_GENERIC_SYSTEM_FIELD=':'2!n[1c]':'"(.*)"`
syntax.GENERIC_FIELD	Defines syntax for a generic field. **Example** `syntax.GENERIC_FIELD=':'2!n1a'::'4!c'/'[8c]'/'"(.*)"`
syntax.GENERIC_CODE_WORD	Defines syntax for parsing out a qualifier, issuer or DSS, code word, and data content of a generic field. In most cases, the code word will be four characters exactly, but there are a few field tags such as 17B that have one letter code words. **Example** `syntax.GENERIC_CODE_WORD=':'4!c'/'[8c]'/'4c"(.*)"`

Property	Description
syntax.HEADER	Defines syntax for a header. **Example** `syntax.HEADER='{'3B':'(".*?")('-}'|'}')`
syntax.CC	Defines syntax for the special function <CC>. **Example** `syntax.CC=2!a`
syntax.CUR	Syntax for the special function <CUR>. **Example** `syntax.CUR=3!a`
syntax.DC_GENERIC	Defines generic syntax for the special function <DC>. **Example** `syntax.DC_GENERIC=1!a`
syntax.DC_SPECIFIC	Defines specific syntax for the special function <DC>. **Example** `syntax.DC_SPECIFIC='D'|'C'`
syntax.DM_GENERIC	Defines generic for the special function <DM>. **Example** `syntax.DM_GENERIC=1!a`
syntax.DM_SPECIFIC	Defines specific syntax for the special function <DM>. **Example** `syntax.DM_SPECIFIC='D'|'M'`
syntax.DATE1	Defines syntax for the special function <DATE1>. **Example** `syntax.DATE1=4!n`
syntax.DATE2	Defines syntax for the special function <DATE2>. **Example** `syntax.DATE2=6!n`
syntax.DATE3	Defines syntax for the special function <DATE3>. **Example** `syntax.DATE3=4!n`
syntax.DATE4	Defines syntax for the special function <DATE4>. **Example** `syntax.DATE4=8!n`
syntax.YEAR	Defines syntax for the special function <YEAR>. **Example** `syntax.YEAR=4!n`
syntax.HHMM	Defines syntax for the special function <HHMM>. **Example** `syntax.HHMM=4!n`

Property	Description
syntax.TIME2	Defines syntax for the special function <TIME2>. **Example** `syntax.TIME2=6!n`
syntax.OFFSET	Defines syntax for the special function <OFFSET>. **Example** `syntax.OFFSET=4!n`
syntax.SWIFTBIC_COPY	Defines syntax for the special function <SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.NONSWIFTBIC_COPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used within the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_COPY=4!a2!a1!c1!c[3!c]`
syntax.SWIFTBIC_NONCOPY	Defines syntax for the special function <SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.SWIFTBIC_NONCOPY=6!a2!c[3!c]`
syntax.NONSWIFTBIC_NONCOPY	Defines syntax for the special function <NON-SWIFTBIC> where it is used outside the context of the copy fields of the common message groups. **Example** `syntax.NONSWIFTBIC_NONCOPY=7!c'1'[3!c]`
syntax.MT	Defines syntax for the special function <MT> (message type). **Example** `syntax.MT=3!n`
syntax.SUB6_GENERIC	Defines generic syntax for the special function <SUB6>. **Example** `syntax.SUB6_GENERIC=1!a3!c`
syntax.SUB6_SPECIFIC	Defines specific syntax for the special function <SUB6>. **Example** `syntax.SUB6_SPECIFIC=('S'3!n)|('N'3!c)|('F'3!c)`
syntax.SBLC	Defines syntax for the special function <SB-LC>. **Example** `syntax.SBLC=(4!a2!c)4!n(4!a2!c)`
syntax.INSTR	Defines syntax for the special function <INSTR>. **Example** `syntax.INSTR=('/'1!c'/'[32x])|('/'2!c'/'[31x])|('/'3!c'/'[30x])|('/'4!c'/'[29x])|('/'5!c'/'[28x])|('/'6!c'/'[27x])|('/'7!c'/'[26x])|('/'8!c'/'[25x])`

Property	Description
syntax.VARSEQU1_GENERIC	Defines generic syntax for the special function <VAR-SEQU-1>. **Example** `syntax.VARSEQU1_GENERIC=34x`
syntax.VARSEQU1_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-1>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU1_SPECIFIC=16x['//'16x]`
syntax.VARSEQU2_GENERIC	Defines generic syntax for the special function <VAR-SEQU-2>. **Example** `syntax.VARSEQU2_GENERIC=21x`
syntax.VARSEQU2_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-2>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU2_SPECIFIC=16x['/'4!x]`
syntax.VARSEQU3_GENERIC	Defines generic syntax for the special function <VAR-SEQU-3>. **Example** `syntax.VARSEQU3_GENERIC=33x`
syntax.VARSEQU3_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-3>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU3_SPECIFIC=16x['/'16x]`
syntax.VARSEQU4_GENERIC	Defines generic syntax for the special function <VAR-SEQU-4>. **Example** `syntax.VARSEQU4_GENERIC=[14x]`
syntax.VARSEQU4_SPECIFIC	Defines specific syntax for the special function <VAR-SEQU-4>. **Note:** The special function will validate the specific syntax. **Example** `syntax.VARSEQU4_SPECIFIC=[4x]['//'8x]`
syntax.ISITC	Defines syntax for the special function <35B/ISITC> that is an ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC='ISIN'1e12!c'CRLF'35x['CRLF'<35B/ISITC/1>]`
syntax.ISITC_NO	Defines syntax for the special function <35B/ISITC> that is a non-ISIN format. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC_NO=(('/'4!a'/'10x'CRLF')|('/'5!a'/'10x'CRLF'))35x['CRLF'<35B/ISITC/2>]`

Property	Description
syntax.ISITC1	Defines syntax for the special function <35B/ISITC/1>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC1=35x['CRLF'35x]0-2`
syntax.ISITC2	Defines syntax for the special function <35B/ISITC/2>. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.ISITC2=35x['CRLF'35x]`
syntax.SIGN_GENERIC	Defines generic syntax for the special function <SIGN>. **Example** `syntax.SIGN_GENERIC=1!x|1!y|1!z`
syntax.SIGN_SPECIFIC	Defines specific syntax for the special function <SIGN>. **Example** `syntax.SIGN_SPECIFIC='+'|'-'`
syntax.IBAN	Defines syntax for the special function <IBAN>. **Example** `syntax.IBAN=2!A2!n1!B[29B]`
syntax.PARTYFLDJ_GENERIC	Defines generic syntax for <PARTYFLD/J>. **Example** `syntax.PARTYFLDJ_GENERIC=40x['CRLF'40x]0-4`
syntax.FLD50F_GENERIC	Defines generic syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_GENERIC=35x`
syntax.FLD50F_SPECIFIC	Defines specific syntax for the special function <FLD50F>. **Example** `syntax.FLD50F_SPECIFIC=('/'34x)|(4!a'/'30x)`
syntax.ERI_CODEWORDS	Defines syntax to validate the ERI code words format. **Example** `syntax.ERI_CODEWORDS=['/OCMT/'<CUR><NUMBER>15'/'['/CHGS/'<CUR><NUMBER>15'/']]`
syntax.ERIF61_GENERIC	Defines syntax to validate the generic format of the <ERI_F61> code word. **Example** `syntax.ERIF61_GENERIC=34x`
syntax.ERIF61_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F61>code word. **Example** `syntax.ERIF61_SPECIFIC=34x`

Property	Description
syntax.ERIFLD72_STRUCTURED_GENERIC	Defines syntax to validate the generic format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_GENERIC=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.ERIFLD72_STRUCTURED_SPECIFIC	Defines syntax to validate the specific format of the <ERI_FLD72structured> code word. **Example** `syntax.ERIFLD72_STRUCTURED_SPECIFIC=210x`
syntax.ERIF72_NARRATIVE_GENERIC	Defines syntax to validate the generic format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_GENERIC=35x['CRLF'35x]0-5`
syntax.ERIF72_NARRATIVE_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F72narrative> code word. **Example** `syntax.ERIF72_NARRATIVE_SPECIFIC=210x`
syntax.ERIF77A_GENERIC	Defines syntax to validate the generic format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_GENERIC=35x['CRLF'35x]0-19`
syntax.ERIF77A_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F77A> code word. **Example** `syntax.ERIF77A_SPECIFIC=700x`
syntax.ERIF79_GENERIC	Defines syntax to validate the generic format of the <ERI_F79> code word. **Example** `syntax.ERIF79_GENERIC=50x['CRLF'50x]0-34`
syntax.ERIF79_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F79> code word. **Example** `syntax.ERIF79_SPECIFIC=1750x`
syntax.ERIF86_GENERIC	Defines syntax to validate the generic format of the <ERI_F86> code word. **Example** `syntax.ERIF86_GENERIC=65x['CRLF'65x]0-5`
syntax.ERIF86_SPECIFIC	Defines syntax to validate the specific format of the <ERI_F86> code word. **Example** `syntax.ERIF86_SPECIFIC=390x`

Property	Description
syntax.F72_OTHER	Defines syntax to validate field 72 where no special function validation is needed. **Example** `syntax.F72_OTHER=35x['CRLF'35x]0-5`
syntax.F79_OTHER	Defines syntax to validate field 79 where no special function validation is needed. **Example** `syntax.F79_OTHER=50x['CRLF'50x]0-34`
syntax.FLD72	Defines specific syntax for <FLD72>. **Example** `syntax.FLD72=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD79	Defines specific syntax for <FLD79>. **Note:** Unlike <FLD72>, the specific syntax does not use the special function <INSTR>. **Example** `syntax.FLD79=50x['CRLF'50x]0-34`
syntax.FLD72_NOT_STP	Defines specific syntax for <FLD72_not_STP>. **Example** `syntax.FLD72_NOT_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP	Defines specific syntax for <FLD72_STP>. **Example** `syntax.FLD72_STP=<INSTR>['CRLF'((<INSTR>)|('//'33x))]0-5`
syntax.FLD72_STP_INS	Defines specific syntax for the /INS format used by <FLD72_STP>. **Example** `syntax.FLD72_STP_INS='/INS/'((<SWIFTBIC>)|(<NON-SWIFTBIC>))`
syntax.REJTRETN72_GENERIC	Defines generic syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_GENERIC=35x['CRLF'35x]0-5`
syntax.REJTRETN72_SPECIFIC	Defines specific syntax for <RETJT/RETN/72>. **Example** `syntax.REJTRETN72_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'(('/'2!c2!n'/')|('/X'1c2!n'/'))[29x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'29x['CRLF''//'33x]0-2]`
syntax.REJTRETN79_GENERIC	Defines generic syntax for <RETJT/RETN/79>. **Example** `syntax.REJTRETN79_GENERIC=50x['CRLF'50x]0-34`

Property	Description
syntax.REJTRETN79_SPECIFIC	Defines specific syntax for <REJT/RETN/79>. **Example** `syntax.REJTRETN79_SPECIFIC=6!x2!n[1a]['/'2c]'CRLF'((‘/’2!c2!n’/’)|(‘/X’1c2!n’/’))[44x]'CRLF''/MREF/'16x['CRLF''/TREF/'16x]['CRLF''/CHGS/'<CUR><AMOUNT>15]['CRLF''/TEXT/'44x['CRLF''//'48x]0-31]`
syntax.SPECIAL_FUNCTION	Defines a SWIFT syntax that defines a single special function. **Note:** The [2n] component covers special functions <AMOUNT> and <NUMBER>. **Example** `syntax.SPECIAL_FUNCTION='<'(50c)'>'[2n]`
syntax.CODEWORD_IRSX	Defines the code word IRSX. **Example** `syntax.CODEWORD_IRSX=IRSX`
syntax.CODEWORD_AMOUNT_NEGATIVE	Defines a negative code word amount. **Example** `syntax.CODEWORD_AMOUNT_NEGATIVE=N`
syntax.CODEWORD_ABIC	Defines the code word /ABIC/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_ABIC=/ABIC/`
syntax.CODEWORD_NAME	Defines the code word /NAME/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NAME=/NAME/`
syntax.CODEWORD_REJT	Defines the code word /REJT/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_REJT=/REJT/`
syntax.CODEWORD_RETN	Defines the code word /RETN/, used for <RETJT/RETN/72> and <REJT/RETN/79> validation. **Example** `syntax.CODEWORD_RETN=/RETN/`
syntax.CODEWORD_ERI	Defines the code word /OCMT/, used for ERI-related validation. **Example** `syntax.CODEWORD_ERI=/OCMT/`
syntax.CODEWORD_INS	Defines the code word /INS/, used for <FLD72_STP> validation. **Example** `syntax.CODEWORD_INS=/INS/`
syntax.CODEWORD_ASCT	Defines the code word /ASCT/, used for <35B/ISITC/2> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ASCT=/ASCT/`

Property	Description
syntax.CODEWORD_NETS	Defines the code word /NETS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_NETS=/NETS/`
syntax.CODEWORD_SSIS	Defines the code word /SSIS/, used for <PARTYFLD/J> validation. **Example** `syntax.CODEWORD_SSIS=/SSIS/`
syntax.CODEWORD_CURR	Defines the code word CURR, that activates validation of rule #101. **Example** `syntax.CODEWORD_CURR=CURR`
syntax.CODEWORD_MONT	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_MONT=MONT`
syntax.CODEWORD_OTHR	Defines the code word for evaluating special validation T75. **Example** `syntax.CODEWORD_OTHR=OTHR`
syntax.CODEWORD_ISIN	Defines the code word for evaluating special validation T12. Code word ISIN is used for <35B/ISITC> validation. **Note:** The ISITC functions are now DEPRECATED. **Example** `syntax.CODEWORD_ISIN=ISIN`
group.PARTYFLDJ.formats	Defines the regex group numbers where code words can be found that require validation. **Example** `group.PARTYFLDJ.formats=4`
group.genericCodeWord.[field_tag].groups	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.groups=1`
group.genericCodeWord.[field_tag].[regex_matching_group_number]	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69C.1=10`
group.genericCodeWord.[field_tag].[regex_matching_group_number].subfield	Defines the regex group numbers where code words can be found that require validation. **Example** `group.genericCodeWord.69D.12.subfield=3`

Property	Description
codeWords.REJTRETN72	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.REJTRETN72='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.REJTRETN79	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.REJTRETN79='/AC01/','/AC02/','/AC03/', '/AC04/','/AC05/','/AC06/','/AM01/','/AM02/', '/AM03/','/AM04/','/AM05/','/AM06/','/AM07/', '/AM08/','/AG01/','/AG02/','/BE01/','/BE02/', '/BE03/','/BE04/','/BE05/','/DT01/','/MS01/', '/PY01/','/RC01/','/RC02/','/RC03/','/RC04/', '/RF01/','/TM01/'`
codeWords.ISITC1	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC1='AAUS','ACED','ACIN','ACMN', 'ACUS','AEUR','ALUX','AQCK','ASED','ASVM','AVAL', 'AWKP','ASICC'`
codeWords.ISITC2	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC2='/ACLT/','/ADIS/','/AEXP/', '/AFLT/','/AIR/','/AMD/','/AOC/','/AOT/','/APN/', '/ASC/','/ASP/','/ASTY/','/ATBA/','/ATK/','/AYLD/' ,'/ASCT/'`
codeWords.ISITC2.ASCT	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.ISITC2.ASCT='BA','CD','CMO','CORP','CP', 'CPP','CS','FHA','FHL','FN','FOR','FUT','GN','GOVT ','IET','MF','MIO','MPO','MPP','MPT','MUNI','OPT', 'PS','RP','RVRP','SL','TD','USTB','WAR','ZOO'`

Property	Description
codeWords.PARTYFLDJ	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.PARTYFLDJ='/ABIC/','/NAME/','/ACCT/', '/ADD1/','/ADD2/','/CITY/','/USFW/','/USCH/', '/GBSC/','/CLRC/','/NETS/','/SSIS/'`
codeWords.PARTYFLDJ.format [format_number]	Defines special function code word lists. **Note:** These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ. **Example** `codeWords.PARTYFLDJ.format2='/NETS/'`

codeWords.rule.[rule_number]	Defines special function code word lists.
	Note: These are code word properties needed for validating special functions such as REJT/RETN/72 or PARTYFLDJ.
	Examples
	`codeWords.rule.5='BEID','TRCO','TESP','MCCO', 'SMDP'`
	`codeWords.rule.257='BEID','TRCO','TESP','MCCO', 'SMDP'`

translator_swift_mp.properties

The translator_swift_mp.properties file is used to set global configuration parameters for SWIFT Market Practices and Fund Templates.

Special Field Exceptions and Required Code Words

The following caveats apply for special field exceptions and required code words:

- The special field exceptions and required code words properties have the following format:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>`

- If a field tag has different validation requirements in different sequences of the same message type, the sequence is appended as follows:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>.<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96, where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `mpField.Exceptions.<mt>[ext].<marketPracticeID>`

- The code word lists are displayed by position and order within the MT. Non-exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>`

- Exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<marketPracticeID>`

- Where a subfield greater than two needs to be validated, the subfield number is appended:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>.<subfield>`
  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`

- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`

- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.

- See the translator_swift.properties file for similar examples. Note that the translator_swift.properties file does not use the **mpField** prefix and it does not use a market practice ID, because it is not applicable for standard SWIFT messages.

Configuration Settings

The following table describes properties used to configure the translator_swift_mp.properties file in the application:

Property	Description
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 1	
mpField.23G.1.1.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.SMPG02='CANC','NEWM':T86`
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 2	
mpField.22F.TRTR.515.SMPG02	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.TRTR.515.SMPG02='TRAD':NODSS:K22`
mpField.90A.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90A.DEAL.515.SMPG02='PRIC':K90`
mpField.90B.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90B.DEAL.515.SMPG02='PRIC':K90`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 1	
mpField.28E.2.1.536.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.28E.2.1.536.IOA001='LAST','MORE','ONLY': T97`
mpField.23G.1.1.515.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.IOA001='NEWM':T86`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 2	

Property	Description
mpField.22F.SFRE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.SFRE.536.IOA001='ADHO','DAIL','INDA','MNTH','WEEK','YEAR','ANNU','SANN':NODSS:K22`
mpField.22F.CODE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.CODE.536.IOA001='COMP':NODSS:K22`

translator_swift_mp_2002.properties

The translator_swift_mp_2002.properties file is used to set global configuration parameters for Market Practices and Fund Templates.

Special Field Exceptions and Required Code Words

The following caveats apply for special field exceptions and required code words:

- The special field exceptions and required code words properties have the following format:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>`

- If a field tag has different validation requirements in different sequences of the same message type, the sequence is appended as follows:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>.<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96, where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `mpField.Exceptions.<mt>[ext].<marketPracticeID>`

- The code word lists are displayed by position and order within the MT. Non-exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>`

- Exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<marketPracticeID>`

- Where a subfield greater than two needs to be validated, the subfield number is appended:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>.<subfield>`
  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`

- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`

- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.

- See the translator_swift.properties file for similar examples. Note that the translator_swift.properties file does not use the **mpField** prefix and it does not use a market practice ID, because it is not applicable for standard SWIFT messages.

Configuration Settings

The following table describes properties used to configure the translator_swift_mp_2002.properties file in the application:

Property	Description
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 1	
mpField.23G.1.1.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.SMPG02='CANC','NEWM':T86`
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 2	
mpField.22F.TRTR.515.SMPG02	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.TRTR.515.SMPG02='TRAD':NODSS:K22`
mpField.90A.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90A.DEAL.515.SMPG02='PRIC':K90`
mpField.90B.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90B.DEAL.515.SMPG02='PRIC':K90`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 1	
mpField.28E.2.1.536.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.28E.2.1.536.IOA001='LAST','MORE','ONLY':T97`
mpField.23G.1.1.515.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.IOA001='NEWM':T86`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 2	
mpField.22F.SFRE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.SFRE.536.IOA001='ADHO','DAIL','INDA','MNTH','WEEK','YEAR','ANNU','SANN':NODSS:K22`

Property	Description
mpField.22F.CODE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.CODE.536.IOA001='COMP':NODSS:K22`

translator_swift_mp_2005.properties

The translator_swift_mp_2005.properties file is used to set global configuration parameters for Market Practices and Fund Templates.

Special Field Exceptions and Required Code Words

The following caveats apply for special field exceptions and required code words:

- The special field exceptions and required code words properties have the following format:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>`

- If a field tag has different validation requirements in different sequences of the same message type, the sequence is appended as follows:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>.<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96, where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `mpField.Exceptions.<mt>[ext].<marketPracticeID>`

- The code word lists are displayed by position and order within the MT. Non-exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>`

- Exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<marketPracticeID>`

- Where a subfield greater than two needs to be validated, the subfield number is appended:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>.<subfield>`
  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`

- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`

- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.

- See the translator_swift.properties file for similar examples. Note that the translator_swift.properties file does not use the **mpField** prefix and it does not use a market practice ID, because it is not applicable for standard SWIFT messages.

Configuration Settings

The following table describes properties used to configure the translator_swift_mp_2005.properties file in the application:

Property	Description
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 1	
mpField.23G.1.1.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.SMPG02='CANC','NEWM':T86`
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 2	
mpField.22F.TRTR.515.SMPG02	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.TRTR.515.SMPG02='TRAD':NODSS:K22`
mpField.90A.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90A.DEAL.515.SMPG02='PRIC':K90`
mpField.90B.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90B.DEAL.515.SMPG02='PRIC':K90`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 1	
mpField.28E.2.1.536.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.28E.2.1.536.IOA001='LAST','MORE','ONLY':T97`
mpField.23G.1.1.515.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.IOA001='NEWM':T86`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 2	
mpField.22F.SFRE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.SFRE.536.IOA001='ADHO','DAIL','INDA','MNTH','WEEK','YEAR','ANNU','SANN':NODSS:K22`

Property	Description
mpField.22F.CODE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.CODE.536.IOA001='COMP':NODSS:K22`

translator_swift_mp_2006.properties

The translator_swift_mp_2006.properties file is used to set global configuration parameters for Market Practices and Fund Templates.

Special Field Exceptions and Required Code Words

The following caveats apply for special field exceptions and required code words:

- The special field exceptions and required code words properties have the following format:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>`

- If a field tag has different validation requirements in different sequences of the same message type, the sequence is appended as follows:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>.<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96, where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `mpField.Exceptions.<mt>[ext].<marketPracticeID>`

- The code word lists are displayed by position and order within the MT. Non-exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>`

- Exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<marketPracticeID>`

- Where a subfield greater than two needs to be validated, the subfield number is appended:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>.<subfield>`
  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`

- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`

- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.

- See the translator_swift.properties file for similar examples. Note that the translator_swift.properties file does not use the **mpField** prefix and it does not use a market practice ID, because it is not applicable for standard SWIFT messages.

Configuration Settings

The following table describes properties used to configure the translator_swift_mp_2006.properties file in the application:

Property	Description
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 1	
mpField.23G.1.1.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.SMPG02='CANC','NEWM':T86`
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 2	
mpField.22F.TRTR.515.SMPG02	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.TRTR.515.SMPG02='TRAD':NODSS:K22`
mpField.90A.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90A.DEAL.515.SMPG02='PRIC':K90`
mpField.90B.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90B.DEAL.515.SMPG02='PRIC':K90`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 1	
mpField.28E.2.1.536.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.28E.2.1.536.IOA001='LAST','MORE','ONLY':T97`
mpField.23G.1.1.515.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.IOA001='NEWM':T86`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 2	
mpField.22F.SFRE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.SFRE.536.IOA001='ADHO','DAIL','INDA','MNTH','WEEK','YEAR','ANNU','SANN':NODSS:K22`

Property	Description
mpField.22F.CODE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.CODE.536.IOA001='COMP':NODSS:K22`

translator_swift_mp_2007.properties

The translator_swift_mp_2007.properties file is used to set global configuration parameters for Market Practices and Fund Templates.

Special Field Exceptions and Required Code Words

The following caveats apply for special field exceptions and required code words:

- The special field exceptions and required code words properties have the following format:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>`

- If a field tag has different validation requirements in different sequences of the same message type, the sequence is appended as follows:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>.<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96, where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `mpField.Exceptions.<mt>[ext].<marketPracticeID>`

- The code word lists are displayed by position and order within the MT. Non-exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>`

- Exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<marketPracticeID>`

- Where a subfield greater than two needs to be validated, the subfield number is appended:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>.<subfield>`
  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`

- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`

- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.

- See the translator_swift.properties file for similar examples. Note that the translator_swift.properties file does not use the **mpField** prefix and it does not use a market practice ID, because it is not applicable for standard SWIFT messages.

Configuration Settings

The following table describes properties used to configure the translator_swift_mp_2007.properties file in the application:

Property	Description
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 1	
mpField.23G.1.1.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.SMPG02='CANC','NEWM':T86`
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 2	
mpField.22F.TRTR.515.SMPG02	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.TRTR.515.SMPG02='TRAD':NODSS:K22`
mpField.90A.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90A.DEAL.515.SMPG02='PRIC':K90`
mpField.90B.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90B.DEAL.515.SMPG02='PRIC':K90`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 1	
mpField.28E.2.1.536.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.28E.2.1.536.IOA001='LAST','MORE','ONLY':T97`
mpField.23G.1.1.515.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.IOA001='NEWM':T86`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 2	
mpField.22F.SFRE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.SFRE.536.IOA001='ADHO','DAIL','INDA','MNTH','WEEK','YEAR','ANNU','SANN':NODSS:K22`

Property	Description
mpField.22F.CODE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.CODE.536.IOA001='COMP':NODSS:K22`

translator_swift_mp_2008.properties

The translator_swift_mp_2008.properties file is used to set global configuration parameters for Market Practices and Fund Templates.

Special Field Exceptions and Required Code Words

The following caveats apply for special field exceptions and required code words:

- The special field exceptions and required code words properties have the following format:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>`

- If a field tag has different validation requirements in different sequences of the same message type, the sequence is appended as follows:

  `mpField.<fieldTag>.<subfield>.<component>.<mt>[ext].<marketPracticeID>.<sequence>`

- If the <mt> component is defined as an asterisk (*), all common group messages apply. For example, n92, n95, n96, where n is a Category 1 through 9.
- Some field tags allow a valid currency code in addition to a code word validation. The CUR keyword triggers this validation, which checks the external SWIFT_Currencies code list.

Code Word Validations for ISO 15022 Messages

The following caveats apply for ISO 15022 messages:

- For each MT the exceptions list is displayed first (if any exceptions exist). The exceptions list has the following format:

  `mpField.Exceptions.<mt>[ext].<marketPracticeID>`

- The code word lists are displayed by position and order within the MT. Non-exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>`

- Exception entries have the following format:

  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<marketPracticeID>`

- Where a subfield greater than two needs to be validated, the subfield number is appended:

  `mpField.<fieldTag>.<qualifier>.<mt>[ext].<marketPracticeID>.<subfield>`
  `mpField.<fieldTag>.<qualifier>.<sequence>.<isoBlock>.<mt>[ext].<subfield>`

- Code words are single-quoted and separated by a comma delimiter. Code word lists have the following format:

  `'<codeWord>'[,'<codeWord>']0-n[:NODSS]:<errorCode>`

- NODSS indicates that the code word validation is only performed if no DSS is present. The <errorCode> will always be the corresponding Kxx code of the field tag.

- See the translator_swift.properties file for similar examples. Note that the translator_swift.properties file does not use the **mpField** prefix and it does not use a market practice ID, because it is not applicable for standard SWIFT messages.

Configuration Settings

The following table describes properties used to configure the translator_swift_mp_2008.properties file in the application:

Property	Description
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 1	
mpField.23G.1.1.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.SMPG02='CANC','NEWM':T86`
Market Practice MT515 Germany: Trade Confirmation - Broker to Asset Manager Section 2	
mpField.22F.TRTR.515.SMPG02	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.TRTR.515.SMPG02='TRAD':NODSS:K22`
mpField.90A.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90A.DEAL.515.SMPG02='PRIC':K90`
mpField.90B.DEAL.515.SMPG02	Defines the codes allowed for the specified field. **Example** `mpField.90B.DEAL.515.SMPG02='PRIC':K90`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 1	
mpField.28E.2.1.536.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.28E.2.1.536.IOA001='LAST','MORE','ONLY':T97`
mpField.23G.1.1.515.IOA001	Defines the codes allowed for the specified field. **Example** `mpField.23G.1.1.515.IOA001='NEWM':T86`
Market Practice MT536 United States: ISITC-IOA NA Reconciliation Working Group Section 2	
mpField.22F.SFRE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.SFRE.536.IOA001='ADHO','DAIL','INDA','MNTH','WEEK','YEAR','ANNU','SANN':NODSS:K22`

Property	Description
mpField.22F.CODE.536.IOA001	Defines the codes allowed for the specified field. NODSS indicates that the code word validation is only performed if no DSS is present. **Example** `mpField.22F.CODE.536.IOA001='COMP':NODSS:K22`

translator_swift_mx.properties

The translator_swift_mx.properties file is used to set global configuration parameters for the following SWIFTNet message types:

- SWIFTNet Cash Reporting version 3.0, 3.1, 3.2, 4.0
- SWIFTNet Exceptions and Investigations 1.0, 1.1, and 1.2
- SWIFTNet Funds 2.0, 2.1, 2.2, 3.0, 3.1, and 4.0
- SWIFTNet Trade Services Utility 1.0

Entries are in the following format:

```
swiftMX.<release>.<message>.<element>[.<attribute>]=<defaultValidationFunct
ion>
[,<parentElement>:<validationFunction>]0-n
```

In the above format, <validationFunction> is one of the following values:

Format	SWIFT Standards Class	Description
BIC	Sw.Stds.D00001	BIC
BEI	Sw.Stds.D00002	BEI
IBAN	Sw.Stds.D00003	IBAN
CC	Sw.Stds.D00004	Country
CURACTIVE	Sw.Stds.D00005	ActiveCurrency
CURACTHIST	Sw.Stds.D00006	ActiveOrHistoricCurrency
AMOUNT	Sw.Stds.D00007	CurrencyAmount
CUR	Sw.Stds.D00007	CurrencyAmount
BICOrBEI	Sw.Stds.D00008	AnyBIC
LANG	Sw.Stds.E00001	LanguageCode
NONE	N/A	Used when no validation should be performed.

Configuration Settings

The following table describes properties used to configure the translator_swift_mx.properties file in the application:

Property	Description
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Cash Reporting 2.0. **Example** `swiftMX.camt.003.001.02.AcctSvcr=BIC`

Property	Description
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Cash Reporting 3.0. **Example** `swiftMX.camt.006.001.03.Ctry=CC`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Cash Reporting 3.1. **Example** `swiftMX.camt.005.001.03.CcyAndAmtRg.Ccy=CUR`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Exceptions and Investigations 1.0. **Example** `swiftMX.camt.007.002.01.Cretr=BICOrBEI`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Exceptions and Investigations 1.1. **Example** `swiftMX.camt.007.002.02.IBAN=IBAN`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Funds 2.0. **Example** `swiftMX.sese.001.001.01.BICOrBEI=BICOrBEI`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Funds 2.1. **Example** `swiftMX.reda.003.001.02.BICOrBEI=BICOrBEI`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Funds 2.2. **Example** `swiftMX.sese.013.001.01.CshCmpnt.Ccy=CUR`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Funds 3.0. **Example** `swiftMX.acmt.002.001.01.CtryOfBirth=CC`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Funds 3.1. **Example** `swiftMX.sese.016.001.01.TtlAmtYrToDt=AMOUNT`
swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Funds 4.0. **Example** `swiftMX.camt.042.001.03.Amt.Ccy=CURACTIVE, CshInBrkdwnDtls:CURACTHIST,ComssnDtls:CURACTHIST, CshOutBrkdwnDtls:CURACTHIST`

swiftMX.[release].[message].[element] [.[attribute]]	Defines properties for SWIFTNet Trade Services Utility 1.0.
	Examples
	`swiftMX.tsmt.009.001.02.BuyrBk=BIC`
	`swiftMX.tsmt.009.001.02.SellrBk=BIC`

ui_swift_message_types.properties

The ui_swift_message_types.properties file is used to set global configuration parameters for SWIFT message types.

Entries are in the following format:

```
MessageType.[message_type].[category].[message]=swift.messagetype.[description]
```

Configuration Settings

The following table describes properties used to configure the ui_swift_message_types.properties file in the application:

Property	Description
MessageType.[message_type].[message_type_number]	Defines valid message types for MT messages. **Example** `MessageType.MT.097=swift.messagetype.MT097`

MessageType.[message_type].[category].[message]	Defines valid message types for SWIFTNet Funds.
	Examples
	`MessageType.Funds.semt.003.001.01=swift.messagetype.semt.003.001.01`
	`MessageType.Funds.sese.009.001.01=swift.messagetype.sese.009.001.01`
	`MessageType.Funds.camt.041.001.02=swift.messagetype.camt.041.001.02`
	`MessageType.Funds.setr.017.001.02=swift.messagetype.setr.017.001.02`
	`MessageType.Funds.reda.003.001.02=swift.messagetype.reda.003.001.02`
	`MessageType.Funds.acmt.002.001.01=swift.messagetype.acmt.002.001.01`

MessageType.[message_type].[category].[message]	Defines valid message types for SWIFTNet Trade Services. **Example** `MessageType.TradeServices.tsmt.001.001.02=swift.messagetype.tsmt.001.001.02`
MessageType.[message_type].[category].[message]	Defines valid message types for SWIFTNet Exceptions and Investigations. **Example** `MessageType.Exceptions.camt.007.002.01=swift.messagetype.camt.007.002.01`
MessageType.[message_type].[category].[message]	Defines valid message types for SWIFTNet Cash Reporting. **Example** `MessageType.CashReporting.camt.003.001.02=swift.messagetype.camt.003.001.02`