# SMPP

## SMPP Component

**CamelSmppFinalStatusAvailable as of Camel 2.2**

This component provides access to an SMSC (Short Message Service Center) over the SMPP protocol to send and receive SMS. The JSMPP library is used for the protocol implementation.

The Camel component currently operates as an ESME (External Short Messaging Entity) and not as an SMSC itself.

The SMS Router project provides an excellent example of an SMS Router daemon based on the Camel framework, routing messages between JMS queues (ActiveMQ) and the SMPP network. Run multiple instances in parallel for high-availability.

Starting with **Camel 2.9** you are also able to execute ReplaceSm, QuerySm, SubmitMulti, CancelSm and DataSm.

Maven users will need to add the following dependency to their `pom.xml` for this component:

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-smpp</artifactId>
    <version>x.x.x</version>
    <!-- use the same version as your Camel core version -->
</dependency>
```

### SMS limitations

SMS is neither reliable or secure. Users who require reliable and secure delivery may want to consider using the XMPP or SIP components instead, combined with a smartphone app supporting the chosen protocol.

- Reliability: although the SMPP standard offers a range of feedback mechanisms to indicate errors, non-delivery and confirmation of delivery it is not uncommon for mobile networks to hide or simulate these responses. For example, some networks automatically send a delivery confirmation for every message even if the destination number is invalid or not switched on. Some networks silently drop messages if they think they are spam. Spam detection rules in the network may be very crude, sometimes more than 100 messages per day from a single sender may be considered spam.
- Security: there is basic encryption for the last hop from the radio tower down to the recipient handset. SMS messages are not encrypted or authenticated in any other part of the network. Some operators allow staff in retail outlets or call centres to browse through the SMS message histories of their customers. Message sender identity can be easily forged. Regulators and even the mobile telephone industry itself has cautioned against the use of SMS in two-factor authentication schemes and other purposes where security is important.

While the Camel component makes it as easy as possible to send messages to the SMS network, it can not offer an easy solution to these problems.

## Data coding, alphabet and international character sets

Data coding and alphabet can be specified on a per-message basis. Default values can be specified for the endpoint. It is important to understand the relationship between these options and the way the component acts when more than one value is set.

Data coding is an 8 bit field in the SMPP wire format.

Alphabet corresponds to bits 0-3 of the data coding field. For some types of message, where a message class is used (by setting bit 5 of the data coding field), the lower two bits of the data coding field are not interpreted as alphabet and only bits 2 and 3 impact the alphabet.

Furthermore, current version of the JSMPP library only seems to support bits 2 and 3, assuming that bits 0 and 1 are used for message class. This is why the Alphabet class in JSMPP doesn't support the value 3 (binary 0011) which indicates ISO-8859-1.

Although JSMPP provides a representation of the message class parameter, the Camel component doesn't currently provide a way to set it other than manually setting the corresponding bits in the data coding field.

When setting the data coding field in the outgoing message, the Camel component considers the following values and uses the first one it can find:

- the data coding specified in a header
- the alphabet specified in a header
- the data coding specified in the endpoint configuration (URI parameter)

Older versions of Camel had bugs in support for international character sets. This feature only worked when a single encoding was used for all messages and was troublesome when users wanted to change it on a per-message basis. Users who require this to work should ensure their version of Camel includes the fix for

**Error rendering macro 'jira'**

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

.

In addition to trying to send the data coding value to the SMSC, the Camel component also tries to analyze the message body, convert it to a Java String (Unicode) and convert that to a byte array in the corresponding alphabet  When deciding which alphabet to use in the byte array, the Camel SMPP component does not consider the data coding value (header or configuration), it only considers the specified alphabet (from either the header or endpoint parameter).

If some characters in the String can't be represented in the chosen alphabet, they may be replaced by the question mark ( ? ) symbol.  Users of the API may want to consider checking if their message body can be converted to ISO-8859-1 before passing it to the component and if not, setting the alphabet header to request UCS-2 encoding.  If the alphabet and data coding options are not specified at all then the component may try to detect the required encoding and set the data coding for you.

The list of alphabet codes are specified in the SMPP specification v3.4, section 5.2.19.  One notable limitation of the SMPP specification is that there is no alphabet code for explicitly requesting use of the GSM 3.38 (7 bit) character set.  Choosing the value 0 for the alphabet selects the SMSC *default* alphabet, this usually means GSM 3.38 but it is not guaranteed.  The SMPP gateway Nexmo actually allows the default to be mapped to any other character set with a control panel option. It is suggested that users check with their SMSC operator to confirm exactly which character set is being used as the default.

## Message splitting and throttling

After transforming a message body from a String to a byte array, the Camel component is also responsible for splitting the message into parts (within the 140 byte SMS size limit) before passing it to JSMPP.  This is completed automatically.

If the GSM 3.38 alphabet is used, the component will pack up to 160 characters into the 140 byte message body.  If an 8 bit character set is used (e.g. ISO-8859-1 for western Europe) then 140 characters will be allowed within the 140 byte message body.  If 16 bit UCS-2 encoding is used then just 70 characters fit into each 140 byte message.

Some SMSC providers implement throttling rules.  Each part of a message that has been split may be counted separately by the provider's throttling mechanism.  The Camel Throttler component can be useful for throttling messages in the SMPP route before handing them to the SMSC.

## URI format

```
smpp://[username@]hostname[:port][?options]
smpps://[username@]hostname[:port][?options]
```

If no **username** is provided, then Camel will provide the default value `smppclient`.
If no **port** number is provided, then Camel will provide the default value `2775`.
**Camel 2.3:** If the protocol name is "smpps", camel-smpp with try to use SSLSocket to init a connection to the server.

You can append query options to the URI in the following format, `?option=value&option=value&...`

## URI Options

| Name | Default Value | Description |
|---|---|---|
| `password` | `password` | Specifies the password to use to log in to the SMSC. |
| `systemType` | `cp` | This parameter is used to categorize the type of ESME (External Short Message Entity) that is binding to the SMSC (max. 13 characters). |
| `dataCoding` | `0` | **Camel 2.11** Defines the data coding according the SMPP 3.4 specification, section 5.2.19. (Prior to **Camel 2.9**, this option is also supported.) Example data encodings are:<br>`0`: SMSC Default Alphabet<br>3: Latin 1 (ISO-8859-1)<br>4: Octet unspecified (8-bit binary)<br>8: UCS2 (ISO/IEC-10646)<br>13: Extended Kanji JIS(X 0212-1990) |
| `alphabet` | `0` | **Camel 2.5** Defines encoding of data according the SMPP 3.4 specification, section 5.2.19. This option is mapped to `Alphabet.java` and used to create the `byte[]` which is send to the SMSC. Example alphabets are:<br>`0`: SMSC Default Alphabet<br>4: 8 bit Alphabet<br>8: UCS2 Alphabet |
| `encoding` | `ISO-8859-1` | **only for SubmitSm, ReplaceSm and SubmitMulti** Defines the encoding scheme of the short message user data. |
| `enquireLinkTimer` | `5000` | Defines the interval in milliseconds between the confidence checks. The confidence check is used to test the communication path between an ESME and an SMSC. |

| | | |
|---|---|---|
| `transactionTimer` | `10000` | Defines the maximum period of inactivity allowed after a transaction, after which an SMPP entity may assume that the session is no longer active. This timer may be active on either communicating SMPP entity (i.e. SMSC or ESME). |
| `initialReconnectDelay` | `5000` | Defines the initial delay in milliseconds after the consumer/producer tries to reconnect to the SMSC, after the connection was lost. |
| `reconnectDelay` | `5000` | Defines the interval in milliseconds between the reconnect attempts, if the connection to the SMSC was lost and the previous was not succeed. |
| `registeredDelivery` | `1` | **only for SubmitSm, ReplaceSm, SubmitMulti and DataSm** Is used to request an SMSC delivery receipt and/or SME originated acknowledgements. The following values are defined:<br>`0`: No SMSC delivery receipt requested.<br>`1`: SMSC delivery receipt requested where final delivery outcome is success or failure.<br>`2`: SMSC delivery receipt requested where the final delivery outcome is delivery failure. |
| `serviceType` | `CMT` | The service type parameter can be used to indicate the SMS Application service associated with the message. The following generic service_types are defined:<br>`CMT`: Cellular Messaging<br>`CPT`: Cellular Paging<br>`VMN`: Voice Mail Notification<br>`VMA`: Voice Mail Alerting<br>`WAP`: Wireless Application Protocol<br>`USSD`: Unstructured Supplementary Services Data |
| `sourceAddr` | `1616` | Defines the address of SME (Short Message Entity) which originated this message. |
| `destAddr` | `1717` | **only for SubmitSm, SubmitMulti, CancelSm and DataSm** Defines the destination SME address. For mobile terminated messages, this is the directory number of the recipient MS. |
| `sourceAddrTon` | `0` | Defines the type of number (TON) to be used in the SME originator address parameters. The following TON values are defined:<br>`0`: Unknown<br>`1`: International<br>`2`: National<br>`3`: Network Specific<br>`4`: Subscriber Number<br>`5`: Alphanumeric<br>`6`: Abbreviated |
| `destAddrTon` | `0` | **only for SubmitSm, SubmitMulti, CancelSm and DataSm** Defines the type of number (TON) to be used in the SME destination address parameters. Same as the `sourceAddrTon` values defined above. |
| `sourceAddrNpi` | `0` | Defines the numeric plan indicator (NPI) to be used in the SME originator address parameters. The following NPI values are defined:<br>`0`: Unknown<br>`1`: ISDN (E163/E164)<br>`2`: Data (X.121)<br>`3`: Telex (F.69)<br>`6`: Land Mobile (E.212)<br>`8`: National<br>`9`: Private<br>`10`: ERMES<br>`13`: Internet (IP)<br>`18`: WAP Client Id (to be defined by WAP Forum) |
| `destAddrNpi` | `0` | **only for SubmitSm, SubmitMulti, CancelSm and DataSm** Defines the numeric plan indicator (NPI) to be used in the SME destination address parameters. Same as the `sourceAddrNpi` values defined above. |
| `priorityFlag` | `1` | **only for SubmitSm and SubmitMulti** Allows the originating SME to assign a priority level to the short message. Four Priority Levels are supported:<br>`0`: Level 0 (lowest) priority<br>`1`: Level 1 priority<br>`2`: Level 2 priority<br>`3`: Level 3 (highest) priority |
| `replaceIfPresentFlag` | `0` | **only for SubmitSm and SubmitMulti** Used to request the SMSC to replace a previously submitted message, that is still pending delivery. The SMSC will replace an existing message provided that the source address, destination address and service type match the same fields in the new message. The following replace if present flag values are defined:<br>`0`: Don't replace<br>`1`: Replace |
| `typeOfNumber` | `0` | Defines the type of number (TON) to be used in the SME. Use the `sourceAddrTon` values defined above. |

| | | |
|---|---|---|
| numberingPlanIndicator | 0 | Defines the numeric plan indicator (NPI) to be used in the SME. Use the sourceAddrNpi values defined above. |
| lazySessionCreation | false | **Camel 2.8 onwards** Sessions can be lazily created to avoid exceptions, if the SMSC is not available when the Camel producer is started.<br>**Camel 2.11 onwards** Camel will check the in message headers 'CamelSmppSystemId' and 'CamelSmppPassword' of the first exchange. If they are present, Camel will use these data to connect to the SMSC. |
| httpProxyHost | null | **Camel 2.9.1:** If you need to tunnel SMPP through a HTTP proxy, set this attribute to the hostname or ip address of your HTTP proxy. |
| httpProxyPort | 3128 | **Camel 2.9.1:** If you need to tunnel SMPP through a HTTP proxy, set this attribute to the port of your HTTP proxy. |
| httpProxyUsername | null | **Camel 2.9.1:** If your HTTP proxy requires basic authentication, set this attribute to the username required for your HTTP proxy. |
| httpProxyPassword | null | **Camel 2.9.1:** If your HTTP proxy requires basic authentication, set this attribute to the password required for your HTTP proxy. |
| sessionStateListener | null | **Camel 2.9.3:** You can refer to a org.jsmpp.session.SessionStateListener in the Registry to receive callbacks when the session state changed. |
| addressRange | "" | **Camel 2.11:** You can specify the address range for the SmppConsumer as defined in section 5.2.7 of the SMPP 3.4 specification. The SmppConsumer will receive messages only from SMSC's which target an address (MSISDN or IP address) within this range. |
| splittingPolicy | ALLOW | **Camel 2.14.1 and 2.15.0:** You can specify a policy for handling long messages:<br>• ALLOW - the default, long messages are split to 140 bytes per message<br>• TRUNCATE - long messages are split and only the first fragment will be sent to the SMSC. Some carriers drop subsequent fragments so this reduces load on the SMPP connection sending parts of a message that will never be delivered.<br>• REJECT - if a message would need to be split, it is rejected with an SMPP NegativeResponseException and the reason code signifying the message is too long. |
| proxyHeaders | null | **Camel 2.17:** These headers will be passed to the proxy server while establishing the connection. |
| maxReconnect | 2147483647 | **Camel 2.18:** Defines the maximum number of attempts to reconnect to the SMSC, if SMSC returns a negative bind response |

You can have as many of these options as you like.

```
smpp://smppclient@localhost:2775?
password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=consumer
```

## Producer Message Headers

The following message headers can be used to affect the behavior of the SMPP producer

| Header | Type | Description |
|---|---|---|
| CamelSmppDestAddr | List/String | **only for SubmitSm, SubmitMulti, CancelSm and DataSm** Defines the destination SME address(es). For mobile terminated messages, this is the directory number of the recipient MS. Is must be a List<String> for SubmitMulti and a String otherwise. |
| CamelSmppDestAddrTon | Byte | **only for SubmitSm, SubmitMulti, CancelSm and DataSm** Defines the type of number (TON) to be used in the SME destination address parameters. Use the sourceAddrTon URI option values defined above. |
| CamelSmppDestAddrNpi | Byte | **only for SubmitSm, SubmitMulti, CancelSm and DataSm** Defines the numeric plan indicator (NPI) to be used in the SME destination address parameters. Use the URI option sourceAddrNpi values defined above. |
| CamelSmppSourceAddr | String | Defines the address of SME (Short Message Entity) which originated this message. |

| | | |
|---|---|---|
| `CamelSmppSourceAddrTon` | `Byte` | Defines the type of number (TON) to be used in the SME originator address parameters. Use the `sourceAddrTon` URI option values defined above. |
| `CamelSmppSourceAddrNpi` | `Byte` | Defines the numeric plan indicator (NPI) to be used in the SME originator address parameters. Use the URI option `sourceAddrNpi` values defined above. |
| `CamelSmppServiceType` | `String` | The service type parameter can be used to indicate the SMS Application service associated with the message. Use the URI option `serviceType` settings above. |
| `CamelSmppRegisteredDelivery` | `Byte` | **only for SubmitSm, ReplaceSm, SubmitMulti and DataSm** Is used to request an SMSC delivery receipt and/or SME originated acknowledgements. Use the URI option `registeredDelivery` settings above. |
| `CamelSmppPriorityFlag` | `Byte` | **only for SubmitSm and SubmitMulti** Allows the originating SME to assign a priority level to the short message. Use the URI option `priorityFlag` settings above. |
| `CamelSmppScheduleDeliveryTime` | `Date` | **only for SubmitSm, SubmitMulti and ReplaceSm** This parameter specifies the scheduled time at which the message delivery should be first attempted. It defines either the absolute date and time or relative time from the current SMSC time at which delivery of this message will be attempted by the SMSC. It can be specified in either absolute time format or relative time format. The encoding of a time format is specified in chapter 7.1.1. in the smpp specification v3.4. |
| `CamelSmppValidityPeriod` | `String/Date` | **only for SubmitSm, SubmitMulti and ReplaceSm** The validity period parameter indicates the SMSC expiration time, after which the message should be discarded if not delivered to the destination. If it's provided as `Date`, it's interpreted as absolute time. **Camel 2.9.1 onwards:** It can be defined in absolute time format or relative time format if you provide it as `String` as specified in chapter 7.1.1 in the smpp specification v3.4. |
| `CamelSmppReplaceIfPresentFlag` | `Byte` | **only for SubmitSm and SubmitMulti** The replace if present flag parameter is used to request the SMSC to replace a previously submitted message, that is still pending delivery. The SMSC will replace an existing message provided that the source address, destination address and service type match the same fields in the new message. The following values are defined:<br>`0`: Don't replace<br>`1`: Replace |
| `CamelSmppAlphabet` / `CamelSmppDataCoding` | `Byte` | **Camel 2.5 For SubmitSm, SubmitMulti and ReplaceSm** (Prior to **Camel 2.9** use `CamelSmppDataCoding` instead of `CamelSmppAlphabet`.) The data coding according to the SMPP 3.4 specification, section 5.2.19. Use the URI option `alphabet` settings above. |
| `CamelSmppOptionalParameters` | `Map<String, String>` | **Deprecated and will be removed in Camel 2.13.0/3.0.0**<br>**Camel 2.10.5 and 2.11.1 onwards and only for SubmitSm, SubmitMulti and DataSm** The optional parameters send back by the SMSC. |
| `CamelSmppOptionalParameter` | `Map<Short, Object>` | **Camel 2.10.7 and 2.11.2 onwards and only for SubmitSm, SubmitMulti and DataSm** The optional parameter which are send to the SMSC. The value is converted in the following way:<br>`String` -> `org.jsmpp.bean.OptionalParameter.COctetString`<br>`byte[]` -> `org.jsmpp.bean.OptionalParameter.OctetString`<br>`Byte` -> `org.jsmpp.bean.OptionalParameter.Byte`<br>`Integer` -> `org.jsmpp.bean.OptionalParameter.Int`<br>`Short` -> `org.jsmpp.bean.OptionalParameter.Short`<br>`null` -> `org.jsmpp.bean.OptionalParameter.Null` |
| CamelSmppEncoding | String | **Camel 2.14.1 and Camel 2.15.0 onwards and only for SubmitSm, SubmitMulti and DataSm**. Specifies the encoding (character set name) of the bytes in the message body. If the message body is a string then this is not relevant because Java Strings are always Unicode. If the body is a byte array then this header can be used to indicate that it is ISO-8859-1 or some other value. Default value is specified by the endpoint configuration parameter *encoding* |
| CamelSmppSplittingPolicy | String | **Camel 2.14.1 and Camel 2.15.0 onwards and only for SubmitSm, SubmitMulti and DataSm**. Specifies the policy for message splitting for this exchange. Possible values are described in the endpoint configuration parameter *splittingPolicy* |

The following message headers are used by the SMPP producer to set the response from the SMSC in the message header

| Header | Type | Description |
|---|---|---|
| `CamelSmppId` | `List<String>/String` | The id to identify the submitted short message(s) for later use. **From Camel 2.9.0**: In case of a ReplaceSm, QuerySm, CancelSm and DataSm this header vaule is a `String`. In case of a SubmitSm or SubmitMultiSm this header vaule is a `List<String>`. |

| | | |
|---|---|---|
| `CamelSmppSentMessageCount` | `Integer` | **From Camel 2.9 onwards only for SubmitSm and SubmitMultiSm** The total number of messages which has been sent. |
| `CamelSmppError` | `Map<String, List<Map<String, Object>>>` | **From Camel 2.9 onwards only for SubmitMultiSm** The errors which occurred by sending the short message (s) the form `Map<String, List<Map<String, Object>>>` (messageID : (destAddr : address, error : errorCode)). |
| `CamelSmppOptionalParameters` | `Map<String, String>` | **Deprecated and will be removed in Camel 2.13.0/3.0.0**<br>**From Camel 2.11.1 onwards only for DataSm** The optional parameters which are returned from the SMSC by sending the message. |
| `CamelSmppOptionalParameter` | `Map<Short, Object>` | **From Camel 2.10.7, 2.11.2 onwards only for DataSm** The optional parameter which are returned from the SMSC by sending the message. The key is the `Short` code for the optional parameter. The value is converted in the following way:<br>`org.jsmpp.bean.OptionalParameter.COctetString` -> `String`<br>`org.jsmpp.bean.OptionalParameter.OctetString` -> `byte[]`<br>`org.jsmpp.bean.OptionalParameter.Byte` -> `Byte`<br>`org.jsmpp.bean.OptionalParameter.Int` -> `Integer`<br>`org.jsmpp.bean.OptionalParameter.Short` -> `Short`<br>`org.jsmpp.bean.OptionalParameter.Null` -> `null` |

## Consumer Message Headers

The following message headers are used by the SMPP consumer to set the request data from the SMSC in the message header

| Header | Type | Description |
|---|---|---|
| `CamelSmppSequenceNumber` | `Integer` | **only for AlertNotification, DeliverSm and DataSm** A sequence number allows a response PDU to be correlated with a request PDU. The associated SMPP response PDU must preserve this field. |
| `CamelSmppCommandId` | `Integer` | **only for AlertNotification, DeliverSm and DataSm** The command id field identifies the particular SMPP PDU. For the complete list of defined values see chapter 5.1.2.1 in the smpp specification v3.4. |
| `CamelSmppSourceAddr` | `String` | **only for AlertNotification, DeliverSm and DataSm** Defines the address of SME (Short Message Entity) which originated this message. |
| `CamelSmppSourceAddrNpi` | `Byte` | **only for AlertNotification and DataSm** Defines the numeric plan indicator (NPI) to be used in the SME originator address parameters. Use the URI option `sourceAddrNpi` values defined above. |
| `CamelSmppSourceAddrTon` | `Byte` | **only for AlertNotification and DataSm** Defines the type of number (TON) to be used in the SME originator address parameters. Use the `sourceAddrTon` URI option values defined above. |
| `CamelSmppEsmeAddr` | `String` | **only for AlertNotification** Defines the destination ESME address. For mobile terminated messages, this is the directory number of the recipient MS. |
| `CamelSmppEsmeAddrNpi` | `Byte` | **only for AlertNotification** Defines the numeric plan indicator (NPI) to be used in the ESME originator address parameters. Use the URI option `sourceAddrNpi` values defined above. |
| `CamelSmppEsmeAddrTon` | `Byte` | **only for AlertNotification** Defines the type of number (TON) to be used in the ESME originator address parameters. Use the `sourceAddrTon` URI option values defined above. |
| `CamelSmppId` | `String` | **only for smsc DeliveryReceipt and DataSm** The message ID allocated to the message by the SMSC when originally submitted. |
| `CamelSmppDelivered` | `Integer` | **only for smsc DeliveryReceipt** Number of short messages delivered. This is only relevant where the original message was submitted to a distribution list.The value is padded with leading zeros if necessary. |
| `CamelSmppDoneDate` | `Date` | **only for smsc DeliveryReceipt** The time and date at which the short message reached it's final state. The format is as follows: YYMMDDhhmm. |

| | | |
|---|---|---|
| `CamelSmppStatus` | `DeliveryReceiptState` | **only for smsc DeliveryReceipt:** The final status of the message. The following values are defined:<br>`DELIVRD`: Message is delivered to destination<br>`EXPIRED`: Message validity period has expired.<br>`DELETED`: Message has been deleted.<br>`UNDELIV`: Message is undeliverable<br>`ACCEPTD`: Message is in accepted state (i.e. has been manually read on behalf of the subscriber by customer service)<br>`UNKNOWN`: Message is in invalid state<br>`REJECTD`: Message is in a rejected state |
| `CamelSmppCommandStatus` | `Integer` | **only for DataSm** The Command status of the message. |
| `CamelSmppError` | `String` | **only for smsc DeliveryReceipt** Where appropriate this may hold a Network specific error code or an SMSC error code for the attempted delivery of the message. These errors are Network or SMSC specific and are not included here. |
| `CamelSmppSubmitDate` | `Date` | **only for smsc DeliveryReceipt** The time and date at which the short message was submitted. In the case of a message which has been replaced, this is the date that the original message was replaced. The format is as follows: YYMMDDhhmm. |
| `CamelSmppSubmitted` | `Integer` | **only for smsc DeliveryReceipt** Number of short messages originally submitted. This is only relevant when the original message was submitted to a distribution list.The value is padded with leading zeros if necessary. |
| `CamelSmppDestAddr` | `String` | **only for DeliverSm and DataSm:** Defines the destination SME address. For mobile terminated messages, this is the directory number of the recipient MS. |
| `CamelSmppScheduleDeliveryTime` | `String` | **only for DeliverSm:** This parameter specifies the scheduled time at which the message delivery should be first attempted. It defines either the absolute date and time or relative time from the current SMSC time at which delivery of this message will be attempted by the SMSC. It can be specified in either absolute time format or relative time format. The encoding of a time format is specified in Section 7.1.1. in the smpp specification v3.4. |
| `CamelSmppValidityPeriod` | `String` | **only for DeliverSm** The validity period parameter indicates the SMSC expiration time, after which the message should be discarded if not delivered to the destination. It can be defined in absolute time format or relative time format. The encoding of absolute and relative time format is specified in Section 7.1.1 in the smpp specification v3.4. |
| `CamelSmppServiceType` | `String` | **only for DeliverSm and DataSm** The service type parameter indicates the SMS Application service associated with the message. |
| `CamelSmppRegisteredDelivery` | `Byte` | **only for DataSm** Is used to request an delivery receipt and/or SME originated acknowledgements. Same values as in Producer header list above. |
| `CamelSmppDestAddrNpi` | `Byte` | **only for DataSm** Defines the numeric plan indicator (NPI) in the destination address parameters. Use the URI option `sourceAddrNpi` values defined above. |
| `CamelSmppDestAddrTon` | `Byte` | **only for DataSm** Defines the type of number (TON) in the destination address parameters. Use the `sourceAddrTon` URI option values defined above. |
| `CamelSmppMessageType` | `String` | **Camel 2.6 onwards**: Identifies the type of an incoming message:<br>`AlertNotification`: an SMSC alert notification<br>`DataSm`: an SMSC data short message<br>`DeliveryReceipt`: an SMSC delivery receipt<br>`DeliverSm`: an SMSC deliver short message |
| `CamelSmppOptionalParameters` | `Map<String, Object>` | **Deprecated and will be removed in Camel 2.13.0/3.0.0**<br>**Camel 2.10.5 onwards and only for DeliverSm** The optional parameters send back by the SMSC. |
| `CamelSmppOptionalParameter` | `Map<Short, Object>` | **Camel 2.10.7, 2.11.2 onwards and only for DeliverSm** The optional parameters send back by the SMSC. The key is the `Short` code for the optional parameter. The value is converted in the following way:<br>`org.jsmpp.bean.OptionalParameter.COctetString` -> `String`<br>`org.jsmpp.bean.OptionalParameter.OctetString` -> `byte[]`<br>`org.jsmpp.bean.OptionalParameter.Byte` -> `Byte`<br>`org.jsmpp.bean.OptionalParameter.Int` -> `Integer`<br>`org.jsmpp.bean.OptionalParameter.Short` -> `Short`<br>`org.jsmpp.bean.OptionalParameter.Null` -> `null` |

JSMPP library

See the documentation of the JSMPP Library for more details about the underlying library.

## Exception handling

This component supports the general Camel exception handling capabilities

When an error occurs sending a message with SubmitSm (the default action), the org.apache.camel.component.smpp.SmppException is thrown with a nested exception, org.jsmpp.extra.NegativeResponseException. Call NegativeResponseException.getCommandStatus() to obtain the exact SMPP negative response code, the values are explained in the SMPP specification 3.4, section 5.1.3.
**Camel 2.8 onwards**: When the SMPP consumer receives a `DeliverSm` or `DataSm` short message and the processing of these messages fails, you can also throw a `ProcessRequestException` instead of handle the failure. In this case, this exception is forwarded to the underlying JSMPP library which will return the included error code to the SMSC. This feature is useful to e.g. instruct the SMSC to resend the short message at a later time. This could be done with the following lines of code:

```
from("smpp://smppclient@localhost:2775?
password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=consumer")
  .doTry()
    .to("bean:dao?method=updateSmsState")
  .doCatch(Exception.class)
    .throwException(new ProcessRequestException("update of sms state failed", 100))
  .end();
```

Please refer to the SMPP specification for the complete list of error codes and their meanings.

## Samples

See the SMS Router source code for a full sample application. Consider integrating your application with SMS Router through queues rather than adding the SMPP code directly to your routes, building a more loosely-coupled architecture.

A route which sends an SMS using the Java DSL:

```
from("direct:start")
  .to("smpp://smppclient@localhost:2775?
      password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=producer");
```

A route which sends an SMS using the Spring XML DSL:

```
<route>
  <from uri="direct:start"/>
  <to uri="smpp://smppclient@localhost:2775?
           password=password&amp;enquireLinkTimer=3000&amp;transactionTimer=5000&amp;systemType=producer"/>
</route>
```

A route which receives an SMS using the Java DSL:

```
from("smpp://smppclient@localhost:2775?
password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=consumer")
  .to("bean:foo");
```

A route which receives an SMS using the Spring XML DSL:

```
  <route>
     <from uri="smpp://smppclient@localhost:2775?
               password=password&amp;enquireLinkTimer=3000&amp;transactionTimer=5000&amp;systemType=consumer"/>
     <to uri="bean:foo"/>
  </route>
```

SMSC simulator

If you need an SMSC simulator for your test, you can use the simulator provided by Logica.

## Debug logging

This component has log level **DEBUG**, which can be helpful in debugging problems. If you use log4j, you can add the following line to your configuration:

```
log4j.logger.org.apache.camel.component.smpp=DEBUG
```

## See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started