Tele2 Content Billing Client Protocol
Content Billing Gateway

Version 6.22

History

Issue	Date	Reason for Changes
4.20	2008-09-01	Additional information about IP Number translation
4.21	2008-09-16	Minor corrections
4.22	2008-09-22	Incorporated previous document about ProviderTransactionID
4.23	2008-10-09	Add status codes for Limits, for future use
4.24	2008-10-15	Merge information from separate document concerning ContactInformation for Swedish CAs
4.25	2008-10-22	Add error code information for SOAP request
4.26	2008-11-12	Add additional status code information
4.27	2008-11-21	Minor change in 5.1.1.2
4.28	2009-01-15	Resending of same ProviderTransactionID after timeout expires (10 sec), modified text on TRANSLATEIP and buying process
4.29	2009-02-12	Correct small mistakes; define new Content Types and Status Codes
4.30	2009-02-25	Improved explanation of amount + VAT
4.31	2009-03-10	New Content Types & new Status Codes
4.32	2009-04-03	Details of TranslateIP XML; clarification of XtraData length
4.33	2009-04-06	Reinsert info about ErrorMessageFileds
4.34	2009-07-02	More detailed info about ProviderTransactionID and resending
4.35	2009-08-27	New content type requiring correct customer type to charge
4.36	2009-09-14	New content type
4.37	2009-09-29	Updated info about content description field
4.38	2009-11-27	New methods (Phone Model and IMEI)
4.39	2010-01-15	Clarification of Soap API
4.40	2010-01-18	Updated info about billing status 1004, new content_type 45
4.41	2010-01-26	New content types 46, 47, 48
4.42	2010-02-23	Updated content type 26,30,32 and billing status 43, 997X, 998X
4.43	2010-04-14	Credit functionality (ReferenceID populated with a ProviderTransactionID of a previously charged transaction to credit)
4.44	2010-05-27	More detailed info about resending and ProviderTransactionID . New content type 52 , positioning
4.45	2010-10-05	New content types 53, 81 (checking billing status without charging), New billing status codes 79 – 87
4.46	2011-01-05	New data for Customer Info Function (PRODUCT = INFO)
4.47	2011-01-19	Update customer info
4.48	2011-03-10	New billing status codes 92-93, new content types 54-70
4.49	2011-04-05	Update chapter reference in 6.2
4.50	2011-05-16	Add appendix 1.
4.51	2011-05-16	New content type 71
4.52	2011-05-30	Updated desc
4.53	2011-09-14	Recommendation of waiting at least 7 days before recycling ProviderTransactionID to allow duplicate check clean up. Testnumbers for CBG-SIT
4.54	2011-10-13	New status code for Bercut
4.55	2011-11-08	New content type 72
4.56	2011-11-16	Add more details to status codes 23,24,25 & 95
4.57	2011-11-17	New content type 73
4.58		More detailed description of contentTypes
4.59	2012-01-05	Reserved codes 100-120 for recharge handling
4.60	2012-01-09	Added new status code 96
4.61	2012-01-18	More info about codes 100-120
4.62	2012-01-25	Customer Extended Info, content_type 101 and status codes 121,122,123,124
4.63	2012-01-25	More info about codes 100-120
4.64	2012-02-15	Clarification regarding MMS Content types 10-12.
4.65	2012-02-22	Clarification in CBG billing status 26, to inform customer to do refill.
4.66	2012-03-02	New content types (74, 75) for roaming data bucket.

4.67	2012-03-20	Corrected chapter reference
4.68	2012-04-12	Small changes relating to protocol version number
4.69	2012-04-12	New content_types (77, 78, 79) for operator user token
4.70	2012-04-13	New CBG billing status 125, 126
4.71	2012-04-16	New CBG billing status 127, 128, 129
4.72	2012-04-17	New content type 83 – Prepaid Loan
4.73	2012-04-18	New CBG billing status 130
4.74	2012-04-18	New CBG billing status 131
4.75	2012-05-30	New content type 84 – Travel Insurance
4.76	2012-06-19	Additional instruction for status 46
4.77	2012-06-25	New CBG billing status 132, 133, 134, 135
4.78	2012-07-26	New CBG billing status 136, 137, 138
4.79	2012-07-31	New Content Type 85
4.80	2012-08-15	Added CBG billing status 139,140,141
4.81	2012-08-27	Added CBG billing status 97
4.82	2012-09-06	New content type 86
4.83	2012-09-10	New content type 87
4.84	2012-09-11	New Currency, 16, KZT
4.85	2012-11-28	New CBG billing status 142
4.86	2012-12-05	Change relating to how KAZ currency KZT is handled.
4.87	2012-12-11	Additional change to KZT handling
4.88	2012-12-19	Changed when status 94 resend with new Provider Transaction ID
4.89	2013-01-24	New content type 88, 89
4.90	2013-02-13	New CBG billing status 143
4.91	2013-02-21	New content type 90
4.92	2013-04-25	New Content type 91. New CBG billing status 144.
4.93	2013-06-19	Add currency for Lat EURO
4.94	2013-06-25	New status codes for Google limits
4.95	2013-09-23	New content type 92
4.96	2013-10-23	Google Limits Codes updated
4.97	2013-10-24	Added information for status 26, IN.
4.98	2013-12-10	Added content_type
4.99	2014-02-18	Added content_type 95
5.00	2014-03-14	Added content_type 96, new status code 153 for barring check
5.01	2014-04-14	Added new status code 154 – 159
5.02	2014-04-22	Added content_type
5.03	2014-05-07	Added new status codes 160 – 163, updated content_type 75, 76
5.04	2014-06-26	Added new status codes 164 – 165
5.05	2014-07-04	Updated information in 6.1 Translation of an IP number…..
5.06	2014-11-06	Add currency for Lithuania EURO (EUT)
5.07		
5.08	2015-06-02	Better information for Status 147
5.09		Added new status code 166
5.10	2016-04-22	Added new status codes 167 -172
6.0	2016-05-13	API changes for Token-support
6.01	2016-12-09	Added new status codes 173-176
6.02	2017-02-07	Added new status codes 177-178
6.03	2017-02-14	Added new content types 120-121
6.04	2017-02-22	Added new content types 122-123
6.06	2017-04-24	Added new content_types for Roaming MMS, 200-, 300-, 400- series and to email dest.
6.07	2017-08-10	Added new status code 179
6.08	2017-12-07	Merge docs, get token function
6.09	2017-12-15	Updated example data last page, added new content_type 124
6.10	2018-01-12	Added closing procedure for Translate IP section 2
6.11	2018-05-03	Updated wrong field name in chapter 3.2.5.1
6.12	2018-05-21	New content types (125, 126) for accounting roaming/national data bucket.
6.13	2018-10-10	New content_type 98

6.14	2019-10-14	Added new status codes 180-190
6.15	2019-11-06	Added new status codes 191-192
6.16	2019-11-28	Added new status code 193
6.17	2020-03-10	New content types (127, 128)
6.18	2020-10-03	Update comment for content type 92, new content types 130, 131 and status codes 194, 195
6.19	2021-01-22	Added new status code 196
6.20	2021-05-05	Fixed some typos and fixed document chapter numbers
6.21	2021-05-10	More info about code 108
6.22	2021-08-23	Added new status codes 197, 198, 199, 401

Contents

1. Introduction & Scope

1.1. The CBG Process

- End user contacts ContentProvider for purchase of an item or service.
- ContentProvider contacts Tele2 for charging using SOAP.
- Tele2 checks that the subscriber has sufficient funds for the purchase, makes a charge, and returns the result to the ContentProvider. See also “The Internal CBG Process” below.
- If the ContentProvider receives information that the subscriber has been charged, approves the purchase and deliver the item or service.
- End user receives item or service, and has been charged.

This document deals with the interaction between the ContentProvider and Tele2.

1.2. The Internal CBG Process

This receives data as defined in the request fields and returns data as defined in the response fields.

The response fields consist of a SOAP return code (RC) [typically 200], and a CBG status code in (CBGRESPONSE).

CBGRESPONSE consists of the CBG Internal status code (billing status) [often value 0 for ‘purchase OK’ – all values documented later herein] and a TransactionId (a unique value ALWAYS assigned to a transactions by the internal CBG process).

1.3. Summary of Tasks

This document describes the protocol to be used to communicate with the Tele2 Content Billing Gateway (CBG). The document is aimed for the content providers (CP) or content aggregators (CA) who shall initiate communication with the CBG.

1.4. Prerequisites and General Conditions

To be able to communicate with the CBG, the CA/CP has to sign a contract with Tele2. Tele2 will provide user-ID and password to be used for logging on to the CBG.

1.5. Versions, Versions, Versions

As from now, the *‘Version’* field of the Request Fields must be set to 208.
Where the version is set to 208, tags for ProviderTransactionID, ReferenceId and XtraData shall be populated.

Within the xml tagged data for XtraData, there is a subfield called *‘Version’* of 2 characters. Each version has its own format for the 98 characters of data which constitute the remainder of XtraData.

1.6. WSDL

A WSDL file for the development of the SOAP commands referred to in this document is available, and should be delivered to Content Providers together with this document.

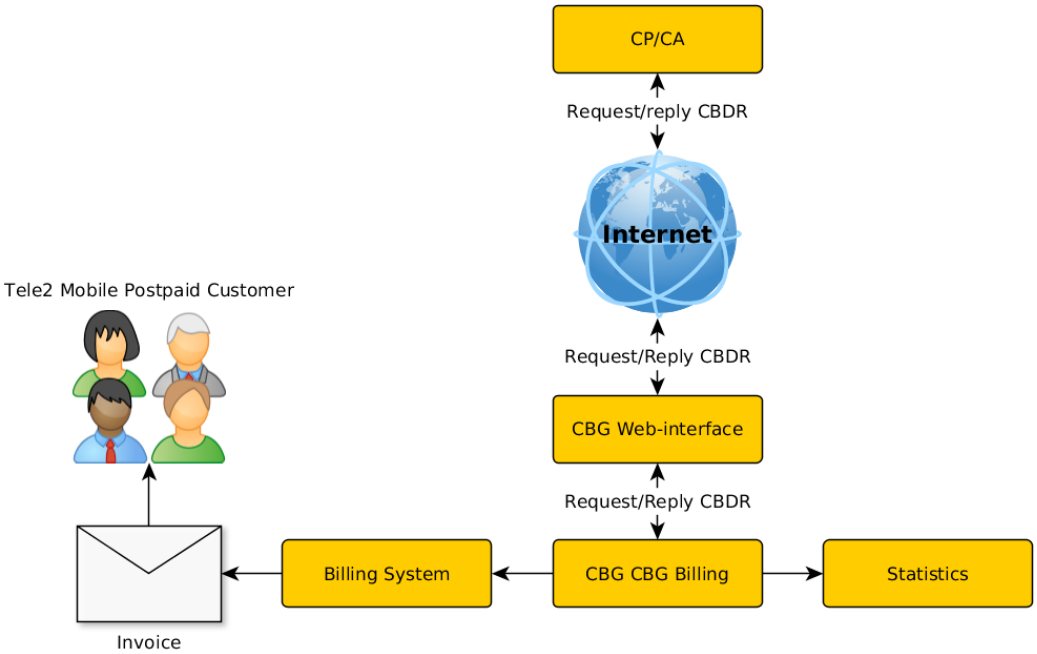
2. Translate IP Method no to be available for external partners

As of March 1st the Translate IP will not be an available Method for content providers (external partners). From March 1st 2018 the Translate IP Method will be in use for Tele2 internal consumers only. At this point this is a target date and will be finally communicated.

Any existing ongoing relationship like a monthly subscription will not be impacted. However, this new get token solution will be the only possible method for any new customers/orders once the closure of translate IP is complete.

3. Interfaces

3.1. Architecture



3.2. Interface Description

3.2.1. Encoding Method

N/A

3.2.2. Transport Method

The transport method for CBG is TCP/IP.

The data is carried using subsets of SOAP 1.1, over HTTPS.

More information regarding these protocol standards can be found at:

SOAP http://soapware.org
SOAP http://www.w3.org/TR/SOAP/
XML http://www.w3.org/TR/2000/REC-xml-20001006
HTTP http://www.w3.org/Protocols/rfc2616/rfc2616.html
SSL http://www.openssl.org

3.2.3. Description

The CA/CP client sends a Request message using this protocol and receives a message in the form of either a Response message or an Error message.

A Response message is sent back for all messages passing basic field control.
A Response message contains a transaction-ID which identifies the transaction in the Content Billing application and it should always be saved by the client along other transaction data for problem resolving etc.
A Response message also contains a status indicator indicating 3 possible outcomes:

- The transaction was committed
- The transaction was rejected. Something in the transaction data did not pass a control, e.g. authentication failed. A request receiving this kind of response should not be resent.
- The transaction failed. The transaction passed controls, but a dynamic fault occurred. A request receiving this kind of response may be resent.

An Error message is sent back when there are problems in the protocol format or whenever there is a communication problem in the backend network.

If you maintain a cookie named “Apache” you may only use that cookie for one specific user (see below). In this case the login.user and login.password fields may be left out for subsequent requests.

3.2.4. To keep in mind

Status 3 or 8 indicates that the A-number is no longer a Tele2 subscriber, and shall be removed from any existing database immediately. Failure to do so will result in charging the wrong subscriber. Non-compliance can in extreme cases be grounds for termination of the CBG Agreement.

Check with the local Tele2 content manager what info should be put in the ContentDescription field. Additional information could be information about the purchased content.

For subscription services the billing requests shall be portioned in order to avoid congestion on the CBG.

ContentType in the billing request shall be set according to access used for delivering the Content.

3.2.5. Data Fields

3.2.5.1. Request Fields

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<Unsigned Int>	yes	208 are supported protocol versions
ContentType	<Unsigned Int>	yes	See. CBG protocol documentation.

Tag	Value	Mandatory	Description
Currency	<Unsigned Int>	yes	See. CBG protocol documentation.
Amount	<Unsigned Int>	yes	See. CBG protocol documentation.
VAT	<Unsigned Int>	yes	See. CBG protocol documentation.
Token	<string>	yes*	Token obtained via the GetToken method
OriginatingCustomerID	<string(5,20)>	yes*	Token from GetToken method or Phone Number in International format
ContentDescription	<string(0,41)>	yes	Content description to be displayed on the customer invoice. Check with local Tele2 content manager how to populate this field. Note, different length for different countries.
ProviderTransactionID	<Unsigned Int>	yes in 208	Values 1-2147483647, then back to 1.
ReferenceID	<Unsigned Int>	yes in 208	0, if normal charging transaction. If a credit transaction, put ProviderTransactionID of a previously charged transaction to credit.
XtraData	<string(0,100)>	yes in 208	See CBG protocol XTRADATA documentation.

* Either Token or OriginatingCustomerID is required, not both.

Instructions for populating these fields:

Tag	Type	Description	
Username	String	Set to the username as given by Tele2	
Password	String	Set to the password as given out by Tele2	
Version	Integer	Should be set to 208. The previous value of 203 will function for a while longer	
ContentType	billing	A type indicator for statistic reports. The only allowed ContentTypes without special approval from Product Manager at Tele2 are:	Functionality triggered by contentType
		0 – MMS Content	Billing
		1 – WAP Content	Billing
		2 – SMS Content/WAP-push	Billing
		3 – MMS Traffic	Billing
		4 – WEB Content	Billing
		5 – Lottery	Billing
		6 – WiFi	Billing
		7 – MO Content	billing
		8 – Personal Ring Back Tone	billing
		9 – Royalty related content	billing
		10 – MMS within the Tele2 network	billing
		11 – MMS to other national operators and e-mail addresses	billing

Tag	Type	Description	
		12 – MMS to international destinations	billing
		13 – MMS to e-mail addresses	billing
		14 – Tickets	billing
		15 – Business	billing
		16 – Age check 16 yrs (Norway only)	Age check + billing
		17 – reserved for special use	billing
		18 – Age check 18 yrs (Norway only)	Age check + billing
		19 – Charity	billing
		20 – Tele2 internal, special billing (0% vat only)	billing
		21 – Parking	billing
		22 – Tele2 internal, mobile IM	billing
		25 – Tele2 internal, special billing (5% vat only)	billing
		26 – Tele2 internal, customer check, no charging	customer info , no billing
		30 – Tele2 internal, customer info PRE/POST, no charging	customer info , no billing
		31 – Mobile Portal Content	billing
		32 – Tele2 internal, limit info, no charging	customer info , no billing
		34 – Tele2 internal, Mobile (Handset) Backup	billing
		35 – Tele2 internal, Limits related Voice	billing
		36 – Tele2 internal, Refill functions related	billing
		38 – Ring Back Tone, charging for prepaid customer type	billing
		39 – Ring Back Tone, charging for postpaid customer type	billing
		40 – Charging for prepaid customer type only	Customer type check + billing
		41 – Charging for postpaid customer type only	Customer type check + billing
		42 – Auction	billing
		43 – Vending machine	billing
		44 – Invoice info	billing
		45 – Tele2 internal, non-content product	billing
		46 – Tele2 internal, Limits related	billing

Tag	Type	Description	
		Data	
		47 – Tele2 internal, Limits related MBB	billing
		48 – Tele2 internal, Limits related SMS	billing
		50 – Adult content (All adult content regardless of type)	billing
		51 – Directory enquiries	billing
		52 – Positioning	billing
		53 – Security product	billing
		54 – Mobile Entertainment (Music, Music videos, Games, Wallpapers, animations, Themes, Videos)	billing
		55 – News (Access to news sites or content bought on a news site)	billing
		56 – Mobile TV (Mobile TV services)	billing
		57 – Applications (software for mobile, any applications that don't fall under any other categories. Includes content bought through a free application (for example an audiobook client))	billing
		58 – Voting (Voting for TV shows or other competitions)	billing
		59 – Betting & gambling	billing
		60 – Classified ads	billing
		61 – Mobile marketing	billing
		62 – Internet Portals (Access to Internet portals that don't fall into news, community or adult)	billing
		63 – Chat (Chatting services, for example dating chats)	billing
		64 – Personal ad (Dating ads, contact ad, dating-sites)	billing
		65 – Community	billing
		66 – Location Based Services	billing
		67 – Information Service (Charging for weather services, snow reports etc.)	billing
		68 – Coupons (For example mobile gift vouchers (ice cream, magazines etc.))	billing
		69 – M-Payment (For example e-school, mobile dictionaries etc.)	billing

Tag	Type	Description	
		70 – Physical goods (Physical goods like, books, drinks, food etc.)	billing
		71 – Tele2 internal, handset down payment	billing
		72 – Tele2 Internal, unlock iPhone fee	billing
		73 – SEB service	billing
		74 – Roaming data bucket	Customer type check + billing
		75 – National data bucket	Customer type check + billing
		76 – National data bucket	Customer type check + billing
		77 – Associate Google user token	no billing
		78 – Check Google user token	customer info, no billing
		79 – Use Google user token	billing
		80 – Test (for CBG SIT environment only)	test (CBG SIT only)
		81 – Check billing status, no charging	customer info, no billing
		82 – Subscription check (not in use)	customer info, no billing
		83 – Prepaid Loan	billing
		84 – Travel Insurance	billing
		85 – GiveMeBalance	Service for balance checking
		86 – Videos STB (Set-top-box)	billing
		87 – Videos STB (Set-top-box), Customer check (no charge)	customer info, no billing
		88 – “+46” Business SE	Billing
		89 – “+46” Residential SE	Billing
		90 – Prepaid daily MB fix	Billing
		91 – Audro PPR Request	Billing
		92 – Special credit	Billing
		93 – Phone number info	Billing
		94 – Collective transport tickets (Norway)	Billing
		95 – Music/Video streaming	Billing
		96 – Media subscription	Billing
		97 – Wholesale charge	Billing
		98 – E-car charges	Billing
		100 – Customer Info Function	customer info, no billing

Tag	Type	Description	
		101 – Customer Extended Info Function	customer info, no billing
		102 – Collective transport tickets + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		103 – Parking + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		104 – Tickets + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		105 – Classified ads + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		106 – Physical goods (Physical goods like, books, drinks, food etc) + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		107 – Phone number info + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		108 – Charity + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		109 – Music/Video streaming + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		110 – Digital Education (Norway only)	billing
		111 – Health Services (Norway only)	billing
		112 – Manual Services (Norway only)	billing
		113 – No Show manual services (Norway only)	billing
		114 – Membership Fee (Norway only)	billing
		115 – Digital Education + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		116 – Health Services + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		117 – Manual Services + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		118 – No Show manual services + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		119 – Membership Fee + Age check 18 yrs (Norway only)	Age check (18 yrs) + billing
		120 – Associate Apple user token	no billing
		121– Use Apple user token	billing
		122– Parking special 1	billing
		123– Parking special 2	billing
		124– Wholesale special	billing

Tag	Type	Description	
		125 – Accounting Roaming data bucket	Customer type check + billing
		126 – Accounting National data bucket	Customer type check + billing
		127 – Mobilly charges (0% vat)	Billing
		128 – Transaction fee Mobilly (21% VAT)	Billing
		130 – Associate operator user token	No billing
		131 – Check operator user token	Customer info, no billing,
		200 – Roaming MO-VAS, MMS sent form a Tele2 subscriber in roaming state to a “Premium” Short Code number with some defined price.	Billing
		210 – Roaming MO -MT, MMS sent from a Tele2 subscriber in roaming state to another Tele2 subscriber. Roaming MO -VAS, MMS sent from a Tele2 subscriber in roaming state to a standard short code . This should be charged as a normal MMS. -	billing
		211 – Roaming MO -MT: MMS sent from a Tele2 subscriber in roaming state to a national subscriber outside the Tele2 network.	Billing
		212 – Roaming MO -MT: MMS sent from a Tele2 subscriber in roaming state to an International number.	Billing
		213 – Roaming MO -Email	Billing

Tag	Type	Description	
		: MMS sent from a Tele2 subscriber in roaming state to an Email.	
		300 – VAS-MT Local : MMS sent form a Short Code to a local Tele2 subscriber	billing
		310 – MO -MT Local : MMS sent from a local Tele2 subscriber to another local Tele2 subscriber.	Billing
		311 – MO -MT Local : MMS sent from a national subscriber outside the Tele2 network to a national Tele2 subscriber.	Billing
		312 – MO -MT Local : MMS sent from an International number to a local Tele2 subscriber.	Billing
		313 – Email -MT Local : MMS sent from an Email to a local Tele2 subscriber.	Billing
		400 – Roaming MO -VAS: MMS sent form a Tele2 subscriber in roaming state to a Short Code recipient.	Billing
		410 – Roaming MO -MT: MMS sent from a Tele2 subscriber in roaming state to another Tele2 subscriber.	Billing

Tag	Type	Description	
		411 – Roaming MO -MT: MMS sent from a Tele2 subscriber in roaming state to a national subscriber outside the Tele2 network.	Billing
		412 – Roaming MO -MT: MMS sent from a Tele2 subscriber in roaming state to an International number.	Billing
		413 – Roaming MO -Email: MMS sent from a Tele2 subscriber in roaming state to an Email.	Billing
		500 series: reserved for internal use.	
		600 series: not in use	
Currency	Integer	A currency indicator. Allowed numeric values are 1 – SEK (Sweden) 2 – NOK (Norway) 3 – DKK (Denmark) 4 – EEK (Estonia) (valid up to and including 2010-12-31) 5 – EUF (Finland) 6 – EUH (Holland) 7 – EUL (Luxembourg) 8 – LVL (Latvia) (valid up to and including 2013-12-31) 9 – LTL (Lithuania) (valid up to and including 2014-12-31) 10 – EUA (Austria) 11 – RUB (Russia) 12 – USD (Russia) 13 – HRK (Croatia) 14 – CHF (Switzerland) 15 – EUE (Estonia) (valid from 2011-01-01) 16 – KZT (Kazakhstan) 17 – EUV (Latvia) (valid from 2014-01-01) 18 – EUT (Lithuania) (valid from 2015-01-01)	
Amount	Integer	**For all countries apart from Kazakhstan:** Price including VAT expressed in local currency subunits, e.g. Swedish öre, Eurocents etc. Ask your local Tele2 product manager. EG. To charge 5 €, set tag “amount” to 500 **However, Kazakhstan is different:** The amount is transmitted in TYIN. This is equivalent to ören or eurocents. The number of TYIN must be “00”, and return code 2 is returned if this rule is violated.	
VAT	Integer	A value in hundreds of percent indicating the VAT. Defaults to 2500 if not set. Ask your local Tele2 Product Manager for correct VAT. EG for 22% VAT, set tag “VAT” to 2200	
Token	String	Token obtained via the GetToken method	

Tag	Type	Description
OriginatingCustomerId	String	If the MSISDN is known that can be used as an argument here. If so, these restrictions apply: MSISDN = CC + NDC + SN (Country Code + National Destination Code + Subscriber Number) Add a “00” International prefix. For example 0046708991199.
ContentDescription	String	Content description to be displayed on the customer invoice. Check with local Tele2 content manager how to populate this field. For Sweden: Max 41 chars. For Estonia, Latvia, Lithuania, Croatia this is restricted to 18 characters – a limitation imposed by the billing system. For Sweden the last 6 characters of the content_description field (static pos 36 (included) – pos 41 (included)) can be used by content_aggregators to populate a code (“contact_id”) referring to a contact_info displayed on the invoice.
ProviderTransactionID	Integer	A unique transaction ID set by the ContentProvider. Start at 1 and increase by 1 for every transaction until 2147483647.After 2147483647, restart at 1
ReferenceID	Integer	0 =>”Normal purchase” or a value referring to a previously sent ProviderTransactionId to be credited.
XtraData	String	Data from the CP. Exactly 100 characters. See definition below.

3.2.5.2. Response Fields

Tag	Value	Description
RC	Signed	Return code 200-600
CBGRESPONSE	Dict	Response from CBG with TransactionID and Status

Note 1:
The ordering of the key-value pairs in the response message is not pre-defined. A Content Provider **can not** assume that the XML tag with key = “RC” will be printed before the tag with key = “CBGRESPONSE”. This means that the Content Provider **must** parse the response message **both** with respect to the **tag/key** and its **value**.

Note 2:
The **only** key-value pairs allowed to implement **automated functionality** on are the ones **documented here** in this paragraph. **Any other** possible key-value pair that is sent inside the response message may be used for logging purposes, but must be treated as **optional and informational only**.

Instructions for interpreting the CBGRESPONSE fields:

Tag	Type	Description
TransactionId	String	Identification of the transaction. Max 30 chars.
Status	Integer	Indication of transaction status. See section 6: ”Billing Status”.

Note:
The key-value pairs inside the CBGRESPONSE dict has the same generic requirements as the key-value pairs in the overall response message. There is no guarantee that the TransactionId tag is printed before the status tag. Also, any other key/tag in this dict response and not defined here in this paragraph must be parsed and treated as optional and informational only (i.e. not used in automated processing but possibly for debugging and logging purposes only).

Additional Instructions for interpreting the CBGRESPONSE fields (for the Customer Info Function):

Tag	Type	Description
TransactionId	String	Identification of the transaction. Max 30 chars.
Status	Integer	Indication of transaction status. See section 6 ”Billing Status”.
Birth	String	Birthday in format yyyy-mm-dd (10 characters)
Gender	String	Gender (6 characters)
PostalNumber	String	Postal number (10 characters)
Xtra1	String	For future use (50 characters)
Xtra2	String	For future use (50 characters)

Note:
The Customer Info Function is activated by the PRODUCT tag containing the String “INFO” at the start. Note that Content Type 100 shall be set, and that Status codes 88 & 89 have special meaning.

3.2.5.3. GetToken Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<Unsigned Int>	yes	203 or 208 are supported protocol versions
OriginatingCustomerIP	<string(7,15)>	yes	IP address on dotted-quad format

3.2.5.4. TranslateIP Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<unsigned int>	yes	208 is supported protocol version
OriginatingCustomerIP	string	yes	IP address or customer identifier from **x-tele2-subid** request header (see section 6.1).

3.2.5.5. Get Phone Model Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<unsigned int>	yes	208 is supported protocol version
Token	string	yes*	Token obtained via the GetToken method
MSISDN	string	yes*	MSISDN = CC + NDC + SN (Country Code + National Destination Code + Subscriber Number)

* Either Token or OriginatingCustomerID is required, not both.

3.2.5.6. Get IMEI Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<unsigned int>	yes	208 is supported protocol version
Token	string	yes*	Token obtained via the GetToken method
MSISDN	string	yes*	MSISDN = CC + NDC + SN (Country Code + National Destination Code + Subscriber Number)

* Either Token or OriginatingCustomerID is required, not both.

3.2.5.7. Get Location Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<unsigned int>	yes	208 is supported protocol version
Token	string	yes*	Token obtained via the GetToken method
MSISDN	string	yes*	MSISDN = CC + NDC + SN (Country Code + National Destination Code + Subscriber Number)

* Either Token or OriginatingCustomerID is required, not both.

3.2.5.8. Error Message Fields

Tag	Type	Description
Rc	nonNegativeInteger	Return Code
error_message	string	Error Message
error_code	string/long/integer	Error Code

4. XtraData Field

4.1. Instructions for coding

Users of CBG shall deliver 100 characters of data in this tag.

Whilst we realize the difficulty of the task, we would nonetheless appreciate it if these 100 characters were populated using the documentation provided.

4.2. XtraData Field Version 01

SubField Name	Type	Description
Version	String	Length 2, Value ‘01’
Delimiter1	String	Length 1, Value ‘;’
ContactID	String	Length 6
Delimiter2	String	Length 1, Value ‘;’
ShortCode	String	Length 20
Delimiter3	String	Length 1, Value ‘;’
Volume	String	Length 6
Delimiter4	String	Length 1, Value ‘;’
DistributionType	String	Length 3
Delimiter5	String	Length 1, Value ‘;’
PurchaseType	String	Length 3
Delimiter6	String	Length 1, Value ‘;’
DeliveryType	String	Length 3
Delimiter7	String	Length 1, Value ‘;’
CategoryType	String	Length 3
Delimiter8	String	Length 1, Value ‘;’
CategorySubType	String	Length 26
Delimiter9	String	Length 1, Value ‘;’
ReasonCode	String	Length 2
Delimiter10	String	Length 1, Value ‘;’
PINCode	String	Length 8
Delimiter11	String	Length 1, Value ‘;’
Unused	String	Length 7

4.3. XtraData Field Version 02

For Tele2 internal use only (specifically MMS).

SubField Name	Type	Description
Version	String	Length 2, Value ‘02’
Filler	String	Length 98, Value undefined Recommend that this is populated with spaces or zeroes.

5. New parameters in protocol 208

5.1. ProviderTransactionID

To increase the transaction security in the purchasing process, a new field ProviderTransactionId has been added in the protocol.

The Content Provider must set a unique ProviderTransactionId in every billing request. Start at value 1 and increase this incrementally by 1 up to 2147483647, after 2147483647, restart at 1.

When content provider receives a CBG reply, check ReturnCode for the parsing result and take action accordingly. If resending, set ProviderTransactionID as stated for the ReturnCode or for the CBG StatusCode in the reply.

If content provider does not receive a CBG reply, content provider can timeout the transaction after 10 sec. Then content provider can resend the transaction with the SAME ProviderTransactionID as the original.

I.e., in case the content provider does not receive a reply from the CBG, the transaction billing status is not known to the content provider, but the transaction can have been billed in the CBG. A resend (after transaction timeout period has expired (10 sec)) with the same ProviderTransactionID (duplicates are saved for at least 24 hours) can be done. If transaction was already processed, the resend will be rejected as a duplicate. Duplicates are checked per account (ContentProviderID). Recommendation is to wait at least 7 days before recycling a ProviderTransactionId to allow for delays in clean-up of the dup check.

Purpose:
The purpose of the ProviderTransactionID is to avoid double charging when the initial transaction failed due to technical problems or reply is lost and the content provider gets no reply on a request (ie status -32400 or it is timeouted at the content provider side). If billing status is unknown the transaction can be re-sent with the same ProviderTransactionId as the original request. If a duplicate ProviderTransactionID is received, CBG will reply status 44, 46 or possibly 999X (depending on the functionality in use), the X will represent the status of the original request with the same ProviderTransactionId. Take action according to the response of the original request.
OBS, if CBG status is received and status advices to increase, a new ProviderTransactionID must be used, otherwise the resending will be regarded as a duplicate.

Example:

1. Original request: ProviderTransactionId = 15 is sent and the content provider gets no reply from CBG (transaction is timed out at content provider side after 10 sec i.e. http error -32400).
2. The Content_provider resends the request after timeout period expired (10 sec) with the SAME ProviderTransactionID = 15.
3. Possible answers:
    a. Reply is status = 9990, == > original request with ProviderTransactionId = 15 was status 0. Purchase OK, customer was charged successfully and content should be delivered.
    b. Reply is status = 9999, == > original request with ProviderTransactionId = 15 was status 9. Customer account balance is to low, reject.
    c. Reply is status = 44, == > original request with ProviderTransactionId = 15 was already processed, but status functionality is not active and original status cannot be shown. Consider as Failed, reject.
    d. Reply is status = 46, == > original request with ProviderTransactionId = 15 was already processed, status functionality is active but could not find original status. Consider as Failed, reject.
    e. No reply, transaction is timed out again at content provider (i.e. http error -32400), == > resend again after timeout period expired (10 sec) with the SAME ProviderTransactionId = 15. Consider as Failed, resend max 3 times. Wait 10 sec between.

Recommended testing:

1. Send a request with a ProviderTransactionId, wait for the reply.
2. Resend request with the same ProviderTransactionId and check that the reply is 44, 46 or 999X.

If you receive 999X, then check that X matches the reply status of the original request.

5.2 Billing status check with ProviderTransactionID

For cases when billing status needs to be checked of a previously sent transaction without re-trying charging, send content_type = 81 and the SAME ProviderTransactionID as previously sent transaction to be checked (for security reasons set amount = 0). CBG will reply status 44, 46, 87 or 999X (depending on the functionality in use), the X will represent the status of the original request with the same ProviderTransactionID.

This can be used for cases when no reply is received from CBG and the billing status is unknown or when customer has required charging approval and billing status is pending customer response.

5.3 ReferenceID

For normal charging, set ReferenceId = 0.
For crediting a previously charged transaction, set ReferenceID to the value of a previously sent ProviderTransactionID to credit (same A-number, content_type etc. should be used as in the previously charged transaction).
For a credited transaction a negative revenue share (based on the credited amount) is added to the content provider revenue share account.

Example:

1. Original charge request: Amount = 100, ProviderTransactionId = 15 AND ReferenceID = 0 is sent, the content provider gets reply Status = 0 == > customer is charged.
2. To credit the charge above, Content_provider sends a request with a new ProviderTransactionID = 16 (increased to a not previously used value) AND ReferenceID = 15 (referring to the charge transaction to credit) and Amount <= 100 (full or partial amount of original charge can be credited, only one credit transaction per original charge transaction is allowed), if content provider gets reply Status = 0 == > customer is credited.

Check with your content manager in what countries the credit function is active. Currently only available for Sweden, Estonia, Latvia, Lithuania, Croatia.

6. Examples using SOAP 1.1 (incl. ProviderTransactionID and XtraData)

6.1.1. Request

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsd=”http://www.w3.org/1999/XMLSchema”
xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <T2api:Call xmlns:T2api=”urn:/T2api/Proto/Soap”>
      <T2api:request>
        <T2api:url>CBG</T2api:url>
        <T2api:method>Purchase</T2api:method>
        <T2api:kwargs>
          <T2api:item>
            <T2api:key>Version</T2api:key>
            <T2api:valueUnsigned>208</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>ContentType</T2api:key>
            <T2api:valueUnsigned>1</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>Currency</T2api:key>
            <T2api:valueUnsigned>1</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>Amount</T2api:key>
            <T2api:valueUnsigned>100</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>VAT</T2api:key>
            <T2api:valueUnsigned>2500</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>OriginatingCustomerId</T2api:key>
            <T2api:valueString>0046704123456</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>username</T2api:key>
            <T2api:valueString>K010101</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>password</T2api:key>
            <T2api:valueString>SecretPassword</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>ContentDescription</T2api:key>
            <T2api:valueString>ProviderDefinedText</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>ProviderTransactionId</T2api:key>
            <T2api:valueUnsigned>1234</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>ReferenceID</T2api:key>
            <T2api:valueUnsigned>0</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>XtraData</T2api:key>
            <T2api:valueString>Provider defined text</T2api:valueString>
          </T2api:item>
        </T2api:kwargs>
      </T2api:request>
    </T2api:Call>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.1.2. Response

Example 1)

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>200</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>rc_message</T2api:key>
          <T2api:valueString>
          </T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>CBGRESPONSE</T2api:key>
          <T2api:valueDict>
            <T2api:item>
              <T2api:key>TransactionId</T2api:key>
              <T2api:valueString>123456789</T2api:valueString>
            </T2api:item>
            <T2api:item>
              <T2api:key>Status</T2api:key>
              <T2api:valueUnsigned>0</T2api:valueUnsigned>
            </T2api:item>
          </T2api:valueDict>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_string</T2api:key>
          <T2api:valueString>Success</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 2)

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>200</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>CBGRESPONSE</T2api:key>
          <T2api:valueDict>
            <T2api:item>
              <T2api:key>TransactionId</T2api:key>
              <T2api:valueString>123456789</T2api:valueString>
            </T2api:item>
            <T2api:item>
              <T2api:key>Status</T2api:key>
              <T2api:valueUnsigned>0</T2api:valueUnsigned>
            </T2api:item>
          </T2api:valueDict>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 3)

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
```

```
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>200</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>CBGRESPONSE</T2api:key>
          <T2api:valueDict>
            <T2api:item>
              <T2api:key>Status</T2api:key>
              <T2api:valueUnsigned>0</T2api:valueUnsigned>
            </T2api:item>
            <T2api:item>
              <T2api:key>TransactionId</T2api:key>
              <T2api:valueString>123456789</T2api:valueString>
            </T2api:item>
          </T2api:valueDict>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_string</T2api:key>
          <T2api:valueString>Success</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_message</T2api:key>
          <T2api:valueString>
          </T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: These are examples. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.3. GetToken XML : Request

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsd=”http://www.w3.org/1999/XMLSchema”
xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <T2api:Call xmlns:T2api=”urn:/T2api/Proto/Soap”>
      <T2api:request>
        <T2api:url>CBG</T2api:url>
        <T2api:method>GetToken</T2api:method>
        <T2api:kwargs>
          <T2api:item>
            <T2api:key>Version</T2api:key>
            <T2api:valueUnsigned>208</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>OriginatingCustomerIP</T2api:key>
            <T2api:valueString>127.1.5.200</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>username</T2api:key>
            <T2api:valueString>K123456</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>password</T2api:key>
            <T2api:valueString>XXXXXXX</T2api:valueString>
          </T2api:item>
        </T2api:kwargs>
      </T2api:request>
    </T2api:Call>
```

```
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.1.4. GetToken XML : (Normal, rc 200) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<soap-env:envelope xmlns:soap-enc=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:soap-
env=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:t2api=”urn:/T2api/Proto/Soap”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”>
  <soap-env:body>
    <t2api:response>
      <t2api:rc>200</t2api:rc>
      <t2api:data>
        <t2api:item>
          <t2api:key>TOKEN</t2api:key>
<t2api:valuestring>UkIv0jElh3KO1SX8bKK2MaJLqOT8O89Ay3Msnb+ykTPVR/eT0xRybEKyZpTSCJfh7iq9maSKeBxqTcbkrbHXbKYA5u
s8fX7N5Zd7L/jdU6E+ymiwGGpEaaOkEw8WXvqr1l8J9zwvIHs9DoAAgyNIIFP+F7WO4CGEcitV09kl1oKe3B2nt71DIA==</t2api:valuest
ring>
        </t2api:item>
      </t2api:data>
    </t2api:response>
  </soap-env:body>
</soap-env:envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.5. GetToken XML : (rc 452) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<soap-env:envelope xmlns:soap-enc=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:soap-
env=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:t2api=”urn:/T2api/Proto/Soap”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”>
  <soap-env:body>
    <t2api:response>
      <t2api:rc>452</t2api:rc>
      <t2api:data>
        <t2api:item>
          <t2api:key>error_code</t2api:key>
          <t2api:valuestring>ObjectNotFound</t2api:valuestring>
        </t2api:item>
        <t2api:item>
          <t2api:key>error_message</t2api:key>
          <t2api:valuestring>Object not found</t2api:valuestring>
        </t2api:item>
      </t2api:data>
    </t2api:response>
  </soap-env:body>
</soap-env:envelope>
```

6.1.6. Get Phone Model : Request

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsd=”http://www.w3.org/1999/XMLSchema”
xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <T2api:Call xmlns:T2api=”urn:/T2api/Proto/Soap”>
      <T2api:request>
        <T2api:url>CBG</T2api:url>
        <T2api:method>GetPhoneModel</T2api:method>
        <T2api:kwargs>
          <T2api:item>
```

```
            <T2api:key>Version</T2api:key>
            <T2api:valueUnsigned>208</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>MSISDN</T2api:key>
            <T2api:valueString>0046704123456</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>username</T2api:key>
            <T2api:valueString>K123456</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>password</T2api:key>
            <T2api:valueString>VerySecretPassword</T2api:valueString>
          </T2api:item>
        </T2api:kwargs>
      </T2api:request>
    </T2api:Call>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.1.7. Return Codes

Return codes are set in the CBG webinterface and indicate if parsing of message was successful (200 = successful parsing), if so also the CBGRESPONSE part in the reply will be populated with CBG Status code (billing status) and CBG TransactionID (Tele2 billing TransactionID), see chapter 6.

When content provider receives a CBG reply, check ReturnCode for the parsing result and take action accordingly. If resending, set ProviderTransactionID as stated for the ReturnCode or for the CBG StatusCode in the reply.

If content provider does not receive a CBG reply, content provider can timeout the transaction after 10 sec. Then content provider can resend the transaction with the SAME ProviderTransactionID as the original.

HTTP Return Codes	Web Standard Meaning	Content Provider Action
1XX	Informational Return Codes	
2XX	Successful Return Codes	
3XX	Redirection Return Code	
4XX	Client Error Return Codes	
5XX	Server Error Return Codes	

CBG Return Codes	CBG Meaning	Content Provider Action
200	Success	CBGRESPONSE part in the reply will be populated with CBG Status code (billing status). Check billing status in chapter 6.
201	Partial	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
202	Accepted	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
203	Already Done	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
204	Created	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
400	UnknownURI	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
401	ReadOnlyURI	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
402	UnknownMethod	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
410	UnknownKey	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
411	ReadonlyKey	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
412	SubspaceNotAllowed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
413	SubspaceNeeded	Resend max 3 times with the SAME ProviderTransactionID as the original.

CBG Return Codes	CBG Meaning	Content Provider Action
		Investigate reason.
414	UnknownSubspace	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
415	DuplicateKey	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
416	ValueNotSet	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
420	ParameterUnknown	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
421	ParameterNeeded	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
422	ParameterSyntaxError	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
423	ParameterInvalid	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
424	ParameterLengthInvalid	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
425	ParameterIllegalCharacters	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
430	AuthenticationFailed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
431	AuthorizationFailed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
432	Suspended	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
433	Disabled	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
440	ClientNotAuthenticated	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
441	ClientNotAuthorized	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
442	UserNotAuthenticated	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
443	UserNotAuthorized	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
444	NotEncypted	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
450	ValueUndefined	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.

CBG Return Codes	CBG Meaning	Content Provider Action
451	SearchCriteriaTooWide	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
452	ObjectNotFound	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
453	OperationNotAllowed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
500	Unavailable	Resend max 3 times with the SAME ProviderTransactionID as the original
501	Aborted	Resend max 3 times with the SAME ProviderTransactionID as the original
502	ProtocolError	Resend max 3 times with the SAME ProviderTransactionID as the original
503	Cluster	Resend max 3 times with the SAME ProviderTransactionID as the original
510	Declined	Resend max 3 times with the SAME ProviderTransactionID as the original
520	NotConfigured	Resend max 3 times with the SAME ProviderTransactionID as the original
521	NotImplemented	Resend max 3 times with the SAME ProviderTransactionID as the original
530	TransactionFailed: often indicates a syntax error in the XML or incompatibility between XML and WSDL (possible with respect to the defined value). Where this is difficult to resolve, test with the XML from this document, and change only that which needs to be changed.	Resend max 3 times with the SAME ProviderTransactionID as the original
531	OutOfResources	Resend max 3 times with the SAME ProviderTransactionID as the original
532	UnknownSession	Resend max 3 times with the SAME ProviderTransactionID as the original
533	Again	
534	Timeout	Resend max 3 times with the SAME ProviderTransactionID as the original
540	BackendUnavailablePermanently	Resend max 3 times with the SAME ProviderTransactionID as the original
541	BackendUnavailableTemporarily	Resend max 3 times with the SAME ProviderTransactionID as the original

6.1.8. Get Phone Model : (Normal, rc = 200) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>200</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>PHONEMODEL</T2api:key>
          <T2api:valueString>SonyEricsson S500i/SonyEricsson S500c</T2api:valueString>
```

```
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_message</T2api:key>
          <T2api:valueString>
          </T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_string</T2api:key>
          <T2api:valueString>Success</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.9. Get Phone Model : (rc = 452) : Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>452</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>rc_string</T2api:key>
          <T2api:valueString>ObjectNotFound</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_message</T2api:key>
          <T2api:valueString>Subscriber not found</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>error_message</T2api:key>
          <T2api:valueString>Subscriber not found</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>error_code</T2api:key>
          <T2api:valueString>ObjectNotFound</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.10. Get IMEI : Request

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsd=”http://www.w3.org/1999/XMLSchema”
xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <T2api:Call xmlns:T2api=”urn:/T2api/Proto/Soap”>
      <T2api:request>
        <T2api:url>CBG</T2api:url>
        <T2api:method>GetIMEI</T2api:method>
        <T2api:kwargs>
```

```
          <T2api:item>
            <T2api:key>Version</T2api:key>
            <T2api:valueUnsigned>208</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>MSISDN</T2api:key>
            <T2api:valueString>0046704123456</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>username</T2api:key>
            <T2api:valueString>K123456</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>password</T2api:key>
            <T2api:valueString>NEVERshowthispassEVER</T2api:valueString>
          </T2api:item>
        </T2api:kwargs>
      </T2api:request>
    </T2api:Call>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.1.11. Get IMEI : (Normal, rc = 200) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>200</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>IMEI</T2api:key>
          <T2api:valueString>354059020628620F</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_message</T2api:key>
          <T2api:valueString>
          </T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>rc_string</T2api:key>
          <T2api:valueString>Success</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.12. Get IMEI : (rc = 452) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/1999/XMLSchema” xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>452</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>rc_string</T2api:key>
          <T2api:valueString>ObjectNotFound</T2api:valueString>
        </T2api:item>
```

```
        <T2api:item>
          <T2api:key>rc_message</T2api:key>
          <T2api:valueString>Subscriber not found</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>error_message</T2api:key>
          <T2api:valueString>Subscriber not found</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>error_code</T2api:key>
          <T2api:valueString>ObjectNotFound</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.13. Translate IP XML : Request

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsd=”http://www.w3.org/1999/XMLSchema”
xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <T2api:Call xmlns:T2api=”urn:/T2api/Proto/Soap”>
      <T2api:request>
        <T2api:url>CBG</T2api:url>
        <T2api:method>TranslateIP</T2api:method>
        <T2api:kwargs>
          <T2api:item>
            <T2api:key>Version</T2api:key>
            <T2api:valueUnsigned>208</T2api:valueUnsigned>
          </T2api:item>
          <T2api:item>
            <T2api:key>OriginatingCustomerIP</T2api:key>
            <T2api:valueString>123.45.67.87</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>username</T2api:key>
            <T2api:valueString>K010101</T2api:valueString>
          </T2api:item>
          <T2api:item>
            <T2api:key>password</T2api:key>
            <T2api:valueString>SecretPassword</T2api:valueString>
          </T2api:item>
        </T2api:kwargs>
      </T2api:request>
    </T2api:Call>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.1.14. Translate IP XML : (Normal, rc 200) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>200</T2api:rc>
      <T2api:data>
```

```
        <T2api:item>
          <T2api:key>OriginatingCustomerId</T2api:key>
          <T2api:valueString>00460704123456</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

6.1.15. Translate IP XML : (rc 452) Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-
ENC=”http://schemas.xmlsoap.org/soap/encoding/”
xmlns:xsi=”http://www.w3.org/1999/XMLSchema-instance” xmlns:xsd=”http://www.w3.org/1999/XMLSchema”
xmlns:T2api=”urn:/T2api/Proto/Soap”>
  <SOAP-ENV:Body>
    <T2api:Response>
      <T2api:rc>452</T2api:rc>
      <T2api:data>
        <T2api:item>
          <T2api:key>error_message</T2api:key>
          <T2api:valueString>ObjectNotFound</T2api:valueString>
        </T2api:item>
        <T2api:item>
          <T2api:key>error_code</T2api:key>
          <T2api:valueString>ObjectNotFound</T2api:valueString>
        </T2api:item>
      </T2api:data>
    </T2api:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.2. Error Scenarios

6.2.1. FAQ

Question:
Do I need an application to connect to CBG?
Answer:
No, you cannot ”connect” to CBG. You send a XML-request over SSL, (HTTPS) to the web service and receive a reply in XML over SSL.

Question:
I cannot find the URL to send the billing requests to in this specification.
Answer:
You will receive it together with your Content Provider ID (CPID) and Password by ordinary mail.

Question:
I receive error code 3 ”Customer Does Not Exist”. I am sure the user is a Comviq/Tele2 customer.
Answer:
The correct format if the customer is ”00 Country Code Mobile Phone number”.
E.g. if the user has mobile number 0704000000 the correct format in the request is 0046704000000

Question:
I receive error code 3, ”Customer Does Not Exist”. I am ***not*** sure the user is a Comviq/Tele2 customer.
Answer:
CBG is not intended to be used for identifying a customer’s operator. Any request to CBG receiving a “Customer does not exist”-reply should be followed by an immediate deletion of the A-number from any

existing customer database. The reason for this is the possible risk of charging new Tele2 customers with A-numbers previously owned by users requesting the content. This risk is only likely to happen for subscription services.

Question:
I cannot access the CBG URL with my web browser. Why is that?
Answer:
You have to use XML and not HMTL which means that a web browser will not suffice. Also remember to use POST instead of GET.

Question:
I cannot access the web service and I'm sure that I'm using XML over SSL. I do use POST and not GET.
Answer:
Please check User-ID and password.

Question:
I still cannot communicate with the CBG.
Answer:
Due to limited resources Tele2 can unfortunately not supply implementation support. Please read the manual thoroughly or consider hiring a consultant for the implementation.

Question:
Why do I receive status code 4 "Purchase within time restriction" when I try to charge multiple transactions of the same customer for the same purchase.
Answer:
You are not allowed to do that, for two reasons:
1. The Maximum limit is intended for each purchase and not for each transaction.
2. Tele2 wants to prevent accidental multiple transactions of customer by mistake.

CBG is not intended for billing of services that are priced above the set maximum amount.

Question:
Can I set up Concurrent connections against the Tele2 content billing gateway?
Answer:
The content provider may set up a maximum of 5 connections against the Tele2 content billing gateway. The main reason for this is that this is the only way to send more than a couple of transactions per second. The secondary reason is that some transactions might take up to several seconds to perform, and if the content provider only has one connection towards the gateway, all other transactions would stall during this period. How this technically is done is completely up to the content provider.

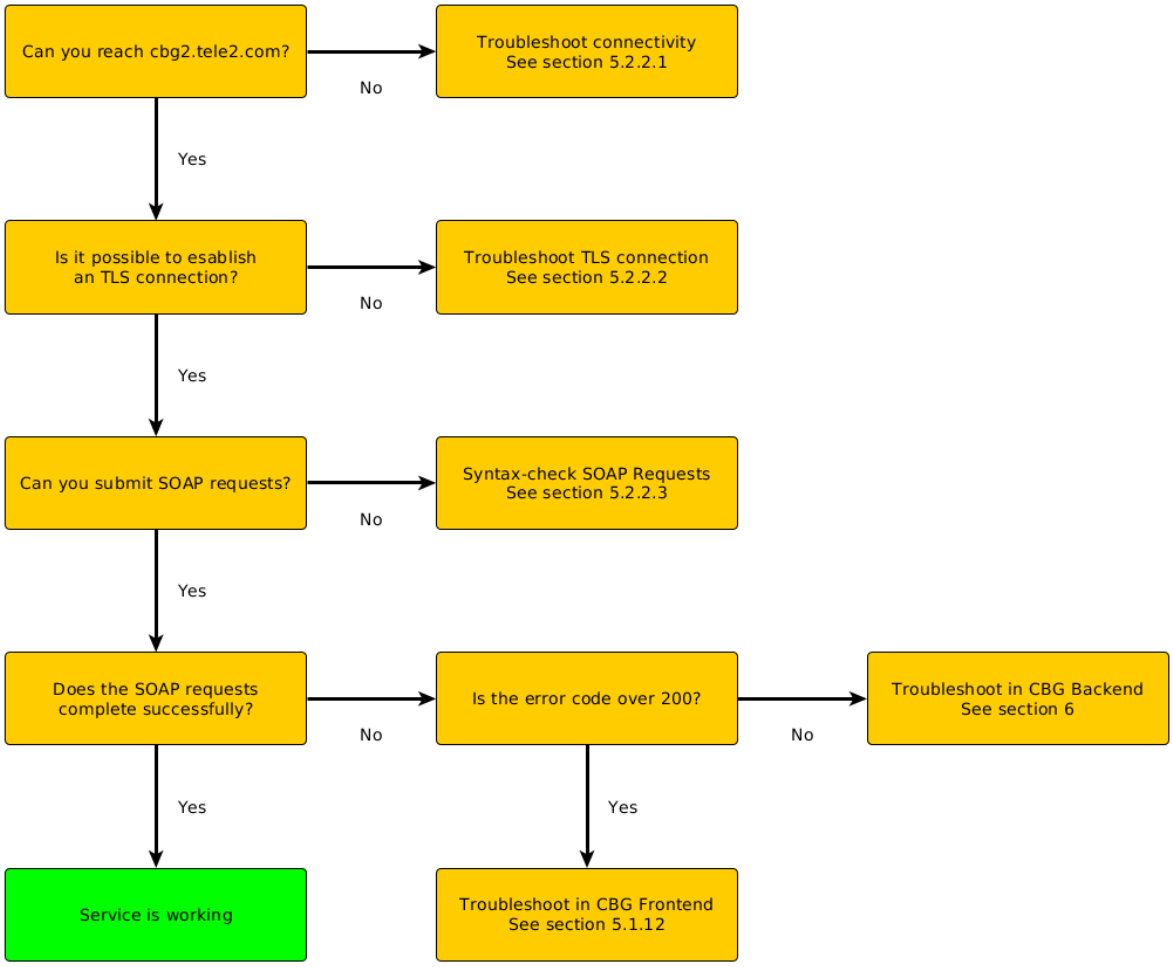
The same session (cookie), can be used for all connections, or they can use different sessions for different connections, or provide the username password in all requests.

6.2.2. Troubleshooting

This section is intended as assistance when troubleshooting issues relating to the CBG system. Whenever troubleshooting issues with the CBG and you have enlist the help of Tele2, please include as much information as possible in the initial contact. This will speed up the handling of your issue. Relevant information in almost all cases include:

- Date and time when the problem occurred
- Source IP of your system (please note that your system might be behind a NAT, we need the public IP of your system or the firewall providing NAT functionality!)
- CBG Account name
- Error messages from your application
- In the case of errors related to return-codes listed in Section 6, include the MSISDN(s) ot Tokens from the transactions that failed.

Blow is a basic flow-chart showing the order in which to investigate issues.



5.2.2.1 Connectivity issues

In order to verify connectivity to the CBG system, suitable tools include “traceroute” and “ping”. Additional insight can be gained using “tcptraceroute”, as it will verify connectivity using TCP on the actual port used by CBG (HTTPS/443).

Example:

```
fli@eastside:~$ tcptraceroute cbg2.tele2.com 443
Selected device eth0, address 212.247.200.180, port 32993 for outgoing packets
Tracing the path to cbg2.tele2.com (212.247.153.186) on TCP port 443 (https), 30 hops max
 170 kst-pe-12.fasteth2-0.swip.net (212.247.200.161)  0.481 ms  0.457 ms  0.421 ms
 2  kst5-core-1.gigabiteth0-7-0-4s14.tele2.net (130.244.39.134)  1.255 ms  0.947 ms  0.612 ms
 3  kst5-spe-1.tengigabiteth3-4.tele2.net (130.244.206.118)  68.843 ms  177.448 ms  1.071 ms
 4  kst5-spe-2.vlan99.tele2.net (130.244.205.186)  58.537 ms  174.665 ms  0.744 ms
 170 cbg2.tele2.com (212.247.153.186) [open]  1.321 ms  1.008 ms  0.761 ms
```

This tells us that in this case the connectivity is working, as indicated by the “open” statement in the final hop.

If you do not have connectivity to the CBG system and the traceroute stops just short of cbg2.tele2.com, please submit a fault report to Tele2. In order for Tele2 to effectively assist you, make sure to include relevant information such as Customer name, CBG Login account and most importantly the source IP-address of your system. Please note that this would be the public IP-address in case your system is behind a NAT and are using RFC1918 private addresses!

5.2.2.2 TLS Issues

A suitable tool for verifying that you are able to establish an SSL connection is the openssl s_client tool. This will establish a secure connection, completing the SSL negotiation and will alert you to any certificate issues. Please note that the CBG system does not support SSLv2 and SSLv3, as well as "weak" crypto algorithms such as DES and RC4. The openssl tool will alert you to many of these problems.

Example:

```
fli@eastside:~$ openssl s_client -connect cbg2.tele2.com:443 -tls1 -no_ssl3

get / HTTP/1.1

HTTP/1.1 405 Method Not Allowed
Server: gSOAP/2.8
Content-Type: text/xml; charset=utf-8
Content-Length: 424
Connection: close
```

The 405 response code from the server means that communication over SSL is possible. Important information when reporting issues to Tele2 relating to SSL negotiation is Customer name, CBG Login account name and Source IP-address. Providing packet-traces, such as produced by "tcpdump" or "Wireshark" for example, of the connection attempt is also helpful.

5.2.2.3 SOAP Issues

When you have verified that the connectivity is OK, it should be possible to submit SOAP-requests. If this is failing, please check the syntax of the requests. There are several tools that can do format validation. One example of an online tool can be found at http://www.xmlvalidation.com/. There are also standalone applications such as SoapUI. SoapUI can also be used to send SOAP requests and is an excellent debugging tool.

The CBG system tries to be specific in its error messages and hints of the error can often be found in the SOAP Response. Please see section 5 for examples of SOAP requests and responses. Section 5.1.12 also includes a list of possible return-codes.

When submitting a fault-report for a SOAP issue it's important to include the failing SOAP-request in its entirety.

7. Billing Status

If message parsing in the CBG web interface was successful (ReturnCode = 200), the CBGRESPONSE part in the reply will be populated with CBG Status code (billing status) and CBG TransactionID (Tele2 billing TransactionID). For a billing to be considered successful (ReturnCode must be 200), CBG status must be 0 and there must be a numeric value of the CBG TransactionID .

Status to CA/CP	StatusDescription	Purchase / Action
0	Purchase OK	OK
1	ContentProvider Does not exist	Reject
2	Amount is out of range (lower or higher than allowed for the content provider account). Or for KZT that the part in TYIN (compare Eurocents) is not zero.	Reject
3	Customer does not exist	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
4	Purchase within time restriction. Customer must not be charged with multiple transactions for the same service.	Reject

Status to CA/CP	StatusDescription	Purchase / Action
5	Invalid routing information	Reject
6	To many requests in timeframe	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
7	To many pending IN requests	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
8	Customer does not exist in IN	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
9	Customer account balance to low	Reject
10	IN read timeout during balance check	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
11	IN read timeout during withdrawal	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
12	Communication error before withdrawal	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
13	Communication error during withdrawal	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
14	Invalid CHARGE ACCOUNT	Reject
15	Invalid VAT_PERCENTAGE	Reject
16	Invalid CURRENCY	Reject
17	IN Node not connected	Reject
18	The content provider account and customer are of different nationalities	Reject
19	Request currency differs from provider account currency	Reject
20	Invalid provider currency. The currency is not registered correctly on the provider account	Reject
21	Max number of outstanding CBG requests exceeded	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
22	Customer has requested barring service for MMS or Content or both MMS and Content.	Reject
23	Internal error: rateMMS_ZONE	Reject
24	Internal error: rateMMS_DAY-TYPE	Reject
25	Internal error: rateMMS_RATE	Reject
26	Customer prepaid account has either expired, or never been activated in IN node for the content service (no first call). For the first case, inform customer to make a refill to activate service, then retry ordering content. For the latter case, inform customer to make a voice call to activate service, then retry ordering content.	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
27	Internal error (From UMTS IN Node)	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
28	Internal error – payment broker	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.

Status to CA/CP	StatusDescription	Purchase / Action
29	Internal error/timeout	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
30	Internal error – transaction already open	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
31	Internal error – transaction busy	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
32	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
33	Internal error – possible overload	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
35	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
36	Customer under 16 yrs	Reject
37	Age check failed (no age data on customer or multiple rows)	Reject
38	Customer under 18 yrs	Reject
39	Customer info prepaid	Reject
40	Customer info prepaid	Reject
41	Customer info error	Reject
42	Customer info postpaid	Reject
43	Customer info : customer active	Reject
44	Duplicate found (ProviderTransactionID already used)	Reject
45	Duplicate function error	Reject
46	Duplicate found (ProviderTransactionID already used) but previous transaction status not found. **Rule1:** The scenario can occur when the call is a resent after a timeout (for example 534 or 541). The timeout usually masks a backend call (usually a call to IN: here the status can be 10 [no money charged] or 11 [money charged, in which case the money will be refunded with the RECHARGE functionality].	Reject **Rule1:** Resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
47	Incoming queue timeout1 Timeout in workflow before call to IN	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
48	Incoming queue timeout2	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
50	Customer has requested barring service for adult content	Reject
51	Customer limit for Content exceeded	Reject
52	Customer limit for MMS exceeded	Reject
53	Customer limit for Volume exceeded	Reject
54	Customer not Active	Reject
55	Customer info – No limit activated	Reject
56	Customer info – No valid limit found	Reject

Status to CA/CP	StatusDescription	Purchase / Action
57	Customer info prepaid	Reject
58	Customer content Limit = 0	Reject
59	Customer info postpaid	Reject
60	Customer info postpaid	Reject
61	Not correct customer type for charging	Failed, customer exists, resend with correct content_type for charging this customer type and new ProviderTransactionID
62	Credit: CREDIT_AMOUNT_LARGER_THAN_CHARGE	Reject Set correct amount
63	Credit: SELECT_CREDIT_STATUS_FAILED	Reject
64	Credit: CREDIT_CONTENTTYPE_DIFFERENT_THAN_CHARGE	Reject Set correct content_type
65	Credit: VAT_DIFFERENT_THAN_CHARGE	Reject Set correct VAT
66	Credit: CURRENCY_DIFFERENT_THAN_CHARGE	Reject Set correct Currency
67	Credit: ORIGINAL_CHARGE_TRANSACTION_NOT_SUCCESSFUL	Reject
68	Credit: CustomerType_DIFFERENT_THAN_CHARGE	Reject
69	Credit: CUSTOMER_DIFFERENT_THAN_CHARGE	Reject Set correct customer
70	Credit: TOO_OLD_TO_CREDIT	Reject
71	Credit: CONTENT_PROVIDER_NOT_ALLOWED_TO_CREDIT	Reject Contact content manager to get permission
72	Credit:	Reject
73	Credit: FAILED_NO_ORIGINAL_CHARGE_FOUND	Reject
74	Credit: CUSTOMER_TYPE_NOT_ALLOWED_TO_CREDIT	Reject
75	Credit: CREDIT_IS_DUPLICATE	Reject
76	Credit: CREDIT_DUPLICATE_CHECK_TABLE_ERROR	Reject
77	Credit: CREDIT_IS_DUPLICATE_SELECT_ORIGINAL_STATUS_FAILED	Reject

Status to CA/CP	StatusDescription	Purchase / Action
78	Credit: CREDIT_IS_DUPLICATE_SELECT_ORIGINAL_PROVIDERTRANSACTIONID_FAILED	Reject
79	Approval: pending customer approval	Pending customer approval, check billing status with new content_type 81 and SAME ProviderTransactionID. Wait 10 sec between resending check.
80	Approval: transaction rejected by customer	Reject
81	Approval: incorrect customer response	Reject
82	Approval: customer response time out	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
83	Approval: approval system error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
84	ProviderTransactionID not correctly populated	Reject
85	Internal error	Reject
86	Dup check: no duplicate found in duplicate transaction cache	Reject
87	Dup check: dup check functionality error	Reject
88	Customer Info Function : Customer Info Found	Reject
89	Customer Info Function : Customer Not Found	Reject
90	Customer Info Function : ContentType not 100	Reject, Set correct content_type
91	Customer Info Function : Table Temporarily Not Available (Try later)	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
92	Database error	Failed, resend max 3 times with SAME ProviderTransactionID. Wait 60 sec between resending.
93	EU Legal requirement, customer registration missing	Reject, Inform customer to register
94	Bercut internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
95	Internal Error: rateMMS: pseudo MSISDN does not exist (e.g. SWEMMS)	Transaction has been rejected. Contact Tele2
96	Internal Error	Reject
97	Content barred based on customer price plan.	Reject
100-120	**Tele2 Internal**	
100	Recharge OK	OK. Either a recharge has functioned OK, or a timed out recharge has been subsequently identified as OK.
101	Recharge Unnecessary	OK
102	Recharge Unsuccessful	NOT OK. Process as specification.
103	Recharge Too Many Timeouts	NOT OK. Process as specification.
104	Recharge Logic Error.	Contact Tele2 Support
105	Recharge Not Implemented for this country / traffic case	Contact Tele2 Support
106	Recharge too old.	NOT OK. Process as specification.
107	Recharge status 28 statistics.	NOT OK. Process as specification.

Status to CA/CP	StatusDescription	Purchase / Action
108	IN getTAState Not Implemented for this country / traffic case	Reject
109	IN recharge : reserved	
110	IN recharge : timeout in check	Process as specification
111	IN recharge : timeout in recharge	Process as specification
112	IN Aggregation Agent : Logic Error	Contact Tele2 Support
113	IN recharge : reserved	
114	IN recharge : reserved	
115	IN recharge : reserved	
116	IN recharge : reserved	
117	IN recharge : reserved	
118	IN recharge : reserved	
119	IN recharge : reserved	
120	IN recharge : reserved	
121	Customer Extended Info Function : Customer Info Found	Reject
122	Customer Extended Info Function : Customer Not Found	Reject
123	Customer Extended Info Function : ContentType not 100	Reject, Set correct content_type
124	Customer Extended Info Function : Table Temporarily Not Available (Try later)	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
125	User token not found	Reject
126	User token expired	Reject
127	Internal error: WFRateDataBucket_GROUP	Reject
128	Internal error: WFRateDataBucket_ZONE	Reject
129	Internal error: WFRateDataBucket_RATE	Reject
130	Internal error: WFRateDataBucket	Reject
131	User token ok	Reject
132	Customer active	Reject
133	Token association not successful	Reject
134	Token table missing	Reject
135	DB error	Reject
136	Not correct currency for converting	Reject
137	User token not populated	Reject
138	User token invalid format	Reject
139	Currency related – zone	Reject
140	Currency related – day type	Reject
141	Currency related – rate	Reject
142	Country not launched in Google	Reject
143	Customer not launched in Google	Reject
144	Customer is not allowed to use premium service	Reject
145	Table missing	Reject. Contact Tele2
146	Row already exists	Reject. Contact Tele2
147	Invalid	Reject. Contact Tele2. Error database handling limit function
148	SQL Error	Reject. Contact Tele2
149		
150		
151	Topup failed – content provider not allowed to topup	Reject
152	Topup failed – RefTransID not = 0	Reject
153	Barring check – customer not barred	Reject

Status to CA/CP	StatusDescription	Purchase / Action
154	KAZ_IN_TIMEOUT_BALANCE_CHECK2	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
155	KAZ_IN_TIMEOUT_WITHDRAWAL2	Reject
156	KAZ_ERROR_BEFORE_WITHDRAWAL2	Reject
157	KAZ_ERROR_DURING_WITHDRAWAL2	Reject
158	KAZ_IN_TIMEOUT_DURING_REFUND	Reject
159	KAZ_IN_COMMUNICATION_FAILURE	Reject
160	KAZ_IN_UNKNOWN_ERROR	Reject
161	KAZ_IN_UNKNOWN_PARAMETER_ERROR	Reject
162	KAZ_IN_UNKNOWN_ERROR2	Reject
163	KAZ_IN_AGGREGATION_ERROR	Reject
164	KAZ_AMOUNT_MUST_END_WITH_00	Reject
165	AGGREGATION_TIMEOUT_FOR_KAZ	Reject
166	WoraPay transaction failed	Reject
167	No MCC, MNC or Zone found for MGT	Reject
168	No valid MCC, MNC or Zone found	Reject
169	No Price plan found	Reject
170	No valid Price plan found	Reject
171	No valid MCC, MNC or Zone found for MGT	Reject
172	No MCC, MNC or Zone found for MGT	Reject
173	Customer does not exist in RPS	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
174	No MCC, MNC found in RPS	Reject
175	RPS Node not connected	Reject
176	Country not launched in Apple	Reject
177	Customer not launched in Apple	Reject
178	RPS Error	Reject
179	Credit: CREDIT_AMOUNT_IS_ALREADY_FULLY_REFUNDED	Reject
180	IN Tis Service Unsuccessful	Reject
181	IN Tis Service Timeout	Reject
185	SV Diameter Unsuccessful	Reject
186	SV Diameter Timeout	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
187	SV Diameter Aggregation Error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
188	Diameter Aggregation Error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
189	Diameter Unknown Error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
190	Total execution timeout	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
191	Diameter_DUPLICATE_FOUND	Reject
192	INTERNAL_SIMULATOR_ERROR	Reject
193	COUPON_CHECK_FAILED	Reject
194	Set Token Association Not Launched	Reject
195	Validate Token Not Launched	Reject

Status to CA/CP	StatusDescription	Purchase / Action
196	Customer Limit updated	No billing
197	Diameter UG Timeout	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
198	Diameter UG Aggregation Error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
199	Diameter UG Unknown Error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
401	Wrong System User Password	Reject. Contact Tele2
1001	Unknown MSISDN	Reject
1002	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
1003	Not Enough Credit	Reject
1004	Not Enough Credit	Reject
1005	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
1006	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
995X	Credit Duplicate found (ReferenceID already used) This status code is used when duplicate_info function is active. Previous transaction status was X (part after the 995 prefix)	Reject Do not resend. See X-part.
997X	Customer has MMS limit and current sum is X (MMS units)	Reject Do not resend. See X-part.
998X	Customer has content limit and current sum is X (subunits of currency, i.e. Eurocents)	Reject Do not resend. See X-part.
999X	Duplicate found (ProviderTransactionID already used). This status code is used when duplicate_info function is active. Previous transaction status was X (part after the 999 prefix)	Reject Do not resend. See X-part.

Note 1. For cases when there are system or network problems, it can happen that a request or reply is lost and the content provider does not receive a reply from the CBG system. Then there might be error codes like:
-32400 or similar. This is an internal XML-RPC error set by the protocol itself at the content provider application side and not sent from the CBG. Action: Investigate your application why this is set. In case it is because of a timeout set on the content provider side, consider the transaction to be failed, resend max 3 times with the SAME ProviderTransactionID (since the previous transaction was probably lost) . Wait 10 sec between resending.
For other XML-RPC error codes like -32600, -32700 etc., see above, these codes are not sent from the CBG, please investigate the application at the content provider side.

Note 2. Billing Status 1001 – 1006 are specific for Russian transactions.

7.1. Translation of an IP number or x-tele2-subid header to a Token

The type of internet access of the subscriber will determine what customer information is available for content billing.

If the subscriber is NATed the source address received by the CP will not be the real end user's address, it will be the address of the NAT device. In the case of a NATed subscriber there will be an additional header in the request, **x-tele2-subid**, containing an identifier for the subscriber. This identifier might be in the form of an IPv4 address, IPv6 address or a string. However, the identifier shall always be treated as a string.

If there is no x-tele2-subid header in the request, the subscriber is not NATed and the getToken method may be called using the source IP-address of the request.

The existing solution for NATed users, the **ip-address** cookie, will remain active for some time. This results in the following process for getting a token to use for the Purchase method:

1. Look for the x-tele2-subid HTTP-header in the request and use its contents for the getToken call.
2. Look for the ip-address cookie in the HTTP request and use its contents for the getToken call.
3. If both the header and the cookie is present, use the contents of the x-tele2-subid for the getToken call.
4. If neither the header of the cookie are present in the request, use the source IPv4 or IPv6 address of the connection for the getToken call.

The next step for the CP is to find out whether the customer is a Tele2/Comviq customer. We exemplify two different ways to do this based on the IP address in examples below (see section 7.2).

The getToken function must be executed separately and before the normal CBG request.

In order to translate an IP address or x-tele2-subid to a Token, the CP performs a similar request as the one to the CBG when asking to charge for content. They use the same login and password and the same URL. The only thing that differs is what arguments they send in the request and which request function they want to use, which is named GetToken (section 3.2.5.3).

Request fields:

Field Name	Type	Description
login.user	String	Same user login as the normal CBG
login.password	String	Password for the account being used
OriginatingCustomerIP	String	The IP address to be for which to get a Token

Output fields:

Field Name	Type	Description
TOKEN	String	The Token to be used in the billing request

7.2. IP Ranges

The IP ranges that Tele2's customers use can change without notice. In order to know if a user is a Tele2 customer or if she/he is a customer to one of the other operators, the IP address has to be looked up in a database. There are two different kinds of databases that could be used for this. Choosing which of these to use is a decision left to the content provider.

The easiest way to perform the lookup is to make a reverse DNS query. If the reverse address contains tele2.xx or swipnet.se, the user is coming from the Tele2 network. For most content providers this would be the easiest way to implement the lookup. There are libraries that support this kind of lookup in virtually all programming environments.

The alternative way of doing the lookup is to make an inquiry to the RIPE database once a day and store all addresses belonging to Tele2 and swipnet in a local database. The RIPE database contains all IP addresses that have been assigned to Tele2. For more information visit http://www.ripe.net.

Currently not all Tele2 customers can be billed through the CBG. If the user is using a dial-up connection, the content provider will not be able to notice that it is not possible to bill the customer through the CBG until the request is sent to the CBG.

For details of XML see 6.1.3.

7.3. Capacity recommendation

Capacity recommendations for sending are:

Max 1 transaction/sec for subscription services and non-capacity requiring services.
Max 1 transaction/sec for the age check functionality for Norway.
Max 5 transactions/sec for live voting services.

8. Appendices

8.1. Appendix 1: To determine whether and MSISDN is Pre- or Post-paid

Submit transaction with ContentType 30.
No charging will take place, but set Amount = 0 for safety reasons.
The Status will be returned as follows:

Decision order	Condition	Return status
1	Postpaid	42
2	Prepaid & ServiceProviderId = LVPDAT	57
3	Prepaid	40
Else		41

8.2. Appendix 2: HTTP Proxy Examples

Example of a HTTP Proxy HTTP Get Request containing the required information:

```
GET / HTTP/1.0
Host: wap.tele2.se
Connection: close
Via: 1.1 tinyproxy (tinyproxy/1.8.3)
Cookie: ip-address=10.69.14.154
Accept: image/vnd.wap.wbmp, image/png, image/jpeg, image/gif, image/bmp, */*
Referer: http://wap.tele2.se/
User-Agent: SAMSUNG-GT-S5610/S5610XXLF3 NetFront/4.1 Profile/MIDP-2.0 Configuration/CLDC-1.1
x-wap-profile: "http://wap.samsungmobile.com/uaprof/GT-S5610_3G.rdf"
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
```

Example of some PHP code how to use an address assigned from a HTTP Get Request sourced from the HTTP Proxy's source IP address:

```
<?php
$hostname = gethostbyaddr($_SERVER['REMOTE_ADDR']);
echo "User agent: " . $_SERVER['HTTP_USER_AGENT'] . "<br>";
echo "Remote address: " . $_SERVER['REMOTE_ADDR'] . "<br>";
echo "Remote host: " . $hostname . "<br><br>";
echo "X-Forwarded-For: " .$_SERVER['X-Forwarded-For'] . "<br>";
if (strpos($hostname,'tele2') !== false) {
    echo "Tele2 customer!";
}

if ($_SERVER['REMOTE_ADDR'] == '90.130.71.165' or $_SERVER['REMOTE_ADDR'] == '90.130.71.164')
{
  echo "<br><br>";
  echo "<h1>Tele2 HTTP-PROXY detectet!</h1>";
  echo "Proxy: " . $_SERVER['HTTP_VIA'] . "<br>";
  echo $_SERVER['HTTP_COOKIE'];
}
?>
```

9. Test numbers for CBG-SIT

Hard coded CBG status codes for testing response handling
Submit transaction to CBG-SIT with OriginatingCustomerID from table below.
A TranslateIP or GetToken on the IP's will return the number in MSISDN column, or in GetToken case a Token that represents this MSISDN.
Then a purchase on the MSISDN/Token will result in the CBG status code listed in the third column.

TranslateIP/GetToken test IP's	MSISDN	CBG status code
127.1.5.200	000000000000	0
127.1.5.1	000000000001	1
127.1.5.2	000000000002	2
127.1.5.3	000000000003	3
127.1.5.4	000000000004	4
127.1.5.6	000000000006	6
127.1.5.7	000000000007	7
127.1.5.8	000000000008	8
127.1.5.9	000000000009	9
127.1.5.10	000000000010	10
127.1.5.11	000000000011	11
127.1.5.12	000000000012	12
127.1.5.13	000000000013	13
127.1.5.22	000000000022	22
127.1.5.26	000000000026	26
127.1.5.28	000000000028	28
127.1.5.33	000000000033	33
127.1.5.35	000000000035	35
127.1.5.37	000000000037	37
127.1.5.42	000000000042	42
127.1.5.47	000000000047	47
127.1.5.51	000000000051	51
127.1.5.54	000000000054	54
127.1.5.56	000000000056	56
127.1.5.61	000000000061	61
127.1.5.70	000000000070	70
127.1.5.73	000000000073	73
127.1.5.79	000000000079	79
127.1.5.100	000000001003	1003
127.1.5.101	000000009950	9950
127.1.5.102	000000099510	99510
127.1.5.103	000000009990	9990
127.1.5.104	000000099910	99910

TranslateIP/GetToken test IP's	MSISDN	CBG status code
n/a	else	Depending on number status in CBG-SIT