nangu.TV IPTV and VoD Platform

Interfaces and API

<help@nangu.tv>

January 20, 2010

Abstract

This document describes interfaces available with nangu.TV VoD and IPTV platform.

Contents

Chapter 1

Provisioning Interface

1.1 Introduction

nangu.TV offers native Web Services (SOAP) calls for interfacing with CRM systems. Web services are invoked through https (HTTP over SSL) connection, where both sides proofs their identity through X.509 certificates. This approach also allows for ISP virtualization, ISP is determined by distinguished name of the caller.

Complementary interface describes options for access billing information in the platform. Billing interface is defined in [1].

1.2 Basic Terms

Subscriber is a platform customer, i.e. an user paying invoices.

Subscription is the service that subscriber is invoiced for. Each subscriber can have more Subscriptions (e.g. subscriber ordered IPTV service for more than one household). Subscriber's subscriptions are completely independent on each other.

Subscription STB account is an account that can be used to connect one set-top box to the platform and recieve the subscribed service. Each subscription can have multiple STB accounts (e.g. multiple set-top boxes in one household). Subscription's STB accounts share same settings (e.g. channel settings, purchase PIN, parental PIN etc.). PUK is unique for each STB account. At most one STB can be logged into one account.

STB is a physical piece of hardware. Each set-top-box is identified by its MAC address and/or serial number. In order to receive IPTV services, set-top box needs to log into the platform using subscription STB account.

Locality is a topology area, which subscriptions belong to. Localities are typically assigned with cache servers (local streamer + storage) and service servers.

SVoD – Subscription VoD is a service where a subscription pays for access of a specific amount of VoD contents on a regular basis (monthly). Access is bought in packages – SVoD packages.

TVoD – Transactional VoD is a service where a subscription pays for rental of specific VoD program on demand, i.e. in time they want to watch it. TVoD categories determine price of programs bought through the TVoD service.

IPTV is a service where a subscription pays for access to a specific amount of IPTV programs on a regular basis (monthly). Access is bought in packages – IPTV packages.

Offer is a container of IPTV and SVOD, packages and other service subscriptions like TVoD and PVR. Offer can be assigned a price in a tariff.

Tariff is a price container for offers. Offers can be assigned different prices in different tariffs.

Subscribers, subscriptions, subscription STB accounts, localities, STB models, SVoD packages, IPTV packages, offers and tariffs are assigned codes used by Service Provider's platform to identify particular instances. These codes are unique for every relation above and have no semantics for nangu.TV platform.

1.3 Provisioning Interface – Client Interface

1.3.1 Data-flows through Provisioning Interface

The following sections describe supported calls. These calls could be interleaved with other actions on Administrator's Portal. The minimal work-flow to enable a subscription is:

1. Subscriber – Create
2. Subscription – Create
3. Subscription – Set Info (sets subscription's PUK)
4. Subscription – Enable

Note: Subscription is created with one STB account by default.

1.3.2 Subscriber – Create

This call is used for creation of a new subscriber in the system. It is invoked by CRM, when nangu.TV service is assigned to the end-customer.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM
- Subscriber's customer number (optional) that can be used for activation of STB

Outputs: Possible data returned by nangu.TV Platform

- ConflictException – request couldn't be fulfilled because the same enabled subscriber exists. The conflict is also raised when the same customer number exists or is not valid (only numbers are enabled) or PUK is bound with customer number and it is not given.

1.3.3 Subscriber – Disable

This call disables a subscriber. It is equal to service removal, but we cannot delete subscriber immediately (we need to keep his/her identity for keeping history). This call is issued by provider's CRM after termination of service contracts with the subscriber.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscriber doesn't exist.
- ConflictException – request couldn't be fulfilled because one or more subscriber's subscriptions are not removed.

1.3.4 Subscriber – Set Info

This call is used for synchronizing additional information about subscriber. This information is useful mainly for helpdesk using the nangu.TV platform. Using this call enables more friendly subscriber and subscription searching and gives helpdesk staff more information about customers.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM.
- Subscriber details (all optional) – customer number, first name, middle name, last name, sex, e-mail, note, billing address consisting of street, city, county, country code and zip code. Setting address to nil value will remove the stored address value.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscriber doesn't exist.
- ConflictException – request couldn't be fulfilled because the same customer number exists or is not valid (only numbers are enabled).

1.3.5 Subscriber – Get Info

This call returns complete information about subscriber.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- Subscriber's identification code
- Subscriber details (all optional) – customer number, first name, middle name, last name, sex, username, e-mail, note, billing address consisting of street, city, county, country code and zip code. Setting address to nil value will remove the stored address value.
- Subscriber create date
- Subscriber subscriptions – same data returned as for `Get Subscriptions` operation.
- NotFoundException – subscriber doesn't exist.

1.3.6 Subscriber – Get Subscriptions

This call returns list of existing subscriber subscriptions and their states.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- List of subscription codes and their states.
- NotFoundException – subscriber doesn't exist.

1.3.7 Subscriber – Search

This call returns list of subscribers corresponded to search parameters. Maximal number of found subscribers is 100 for one request. No parameters filled means "find all subscribers".

Inputs: Data passed towards nangu.TV Platform

- Search parameters (all optional) – search text, search type (field where text should be searched), subscription state, subscription STB account state, locality code, offer code.
- First result – It's usable when real number of found subscribers is greater than 100.

Outputs: Possible data returned by nangu.TV Platform

- Number of found subscribers.
- List of subscribers codes and their full name consisting of first name, middle name and last name. The list is sorted primary by last name and secondary by first name.
- NotFoundException – locality or offer doesn't exist.

1.3.8 Subscription – Create

This call adds a new subscription with one STB account to an existing subscriber. Subscription starts with state NEW and STB account with state STB_ORDERED.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM.
- Subscription's identification code in provider's CRM.
- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, Subscription's code is used.
- Currency code.
- Alternative currency code (optional). Used to display prices also in alternative currency. Must differ from primary currency.
- Initial tariff identification.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscriber, currency, alternative currency or the tariff doesn't exist.
- ConflictException – request couldn't be fulfilled because, alternative currency is same as currency or the same subscription or STB account exists.

1.3.9 Subscription – Create Without Stb Account

This call adds a new subscription without any STB account to an existing subscriber. Subscription starts with state NEW.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscriber's identification code in provider's CRM.
- Subscription's identification code in provider's CRM.
- Currency code.
- Alternative currency code (optional). Used to display prices also in alternative currency. Must differ from primary currency.
- Initial tariff identification.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscriber, currency, alternative currency or the tariff doesn't exist.
- ConflictException – request couldn't be fulfilled because, alternative currency is same as currency or the same subscription exists.

1.3.10 Subscription – Set Info

This call is used for synchronizing additional information about subscription. This information is useful mainly for helpdesk using the nangu.TV platform. Using this call enables more friendly subscriber and subscription searching and gives helpdesk staff more information about customers and provisioning locations.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, one and only one subscription's STB account is used.
- Subscription details (all optional) – PUK, provisioning address and shipping address consisting of street, city, county, country code and zip code; note, allowed IP adresses. Setting an address or PUK to nil value will remove the stored address or PUK value.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – the subscription, STB account or the country doesn't exist
- ConflictException – the same PUK value is already assigned or PUK is not valid number, IP address is not valid or subscription has more than one or has not any STB account whereas STB account code is not defined.

1.3.11 Subscription – Get Info

This call returns complete information about subscription.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription details (mandatory) – subscription code, subscriber code, currency code, tariff code, subscription state, TVOD ordered, parental search, parental detail, create date, nPVR seconds used, maximum nPVR seconds, nPVR history days.
- Subscription details (all optional) – alternative currency code, locality code, enable date, parental PIN, charge limit, note, provisioning address and shipping address consisting of street, city, county, country code and zip code.

Outputs: Possible data returned by nangu.TV Platform

- Subscription details (all mandatory) – subscription code, subscriber code, currency code, tariff code, subscription state, TVOD ordered, parental search, parental detail, create date, nPVR seconds used, maximum nPVR seconds, nPVR history days.

- Subscription details (all optional) – alternative currency code, locality code, enable date, parental PIN, charge limit, note, provisioning address and shipping address consisting of street, city, county, country code and zip code.
- Subscription offers – same data returned as for `Get Offers` operation.
- Subscription STB accounts – same data returned as for `Get Subscription STB Accou` operation.
- NotFoundException – the subscription doesn't exist.

1.3.12 Subscription – Set Locality

This call sets provisioning locality for the subscription.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Locality code.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription or locality doesn't exist.

1.3.13 Subscription – Enable

This call enables activation of an existing subscription and one STB account. Subscription STB account's state will be switched to state STB_SHIPPED (subscription STB account in STB_ACTIVE state is not affected). The subscription's state will be changed to BILLING and STB account's to STB_ACTIVE on the first STB use. If subscription's state is SUSPENDED, subscription's state will be changed to BILLING or NEW (it depends on previous subscription state). If SUSPENDED subscription is enabled and any subscription STB account is in STB_ACTIVE state and its original STB is still assigned to it, user authentication by PUK verification will be skipped.

Subscription and STB account are checked, whether meet the requirements of system config (E.g. valid PUK, filled allowed IPs, assigned STB, filled customer number). All possible required parameters can be set with this call for easier enabling of subscription.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, one and only one subscription's STB account is used.

- Date of enabling the service. Service billing is started automatically on this date. When no date is given or STB is used before this date, the billing is started on first STB usage (if subscription is not suspended). If subscription's state is SUSPENDED and no date is given and at least one STB account in STB_ACTIVE state exists, service billing is started immediately.
- Puk (optional)
- Allowed IP adresses (optional)
- STB to assign (optional). Parameters are the same parameters as for Assign STB request.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription or STB account or STB model doesn't exist.
- ConflictException – subscription or STB account is not ready to be enabled (reason of this is described in the exception), subscription or STB account is already enabled, subscription has more than one or has not any STB account whereas STB account code is not defined, the same PUK value is already assigned, PUK is not valid number or IP address is not valid. Conflict is also raised when the same stb is assigned to another subscription STB account (exception detail contains subscription STB account code of that subscription), or subscription STB account has assigned another STB. Conflict is raised when STB serial number or MAC address exists or multiple STBs have same parameters.

1.3.14 Subscription – Enable with generated PUK

This call generates random PUK and enables activation of an existing subscription and one STB account. Subscription enabling is the same as for Subscription – Enable, whereas PUK is generated instead of explicitly defined.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, one and only one subscription's STB account is used.
- Date of enabling the service.
- Allowed IP adresses (optional)
- STB to assign (optional)
- Requested PUK length.

Outputs: Possible data returned by nangu.TV Platform

- Generated PUK value of given length. It is string with digits from 0 to 9 (e.g. 02975).
- NotFoundException – subscription or STB account doesn't exist.
- ConflictException – subscription or STB account is not ready to be enabled (reason of this is described in the exception), subscription or STB account is already enabled, the PUK length is not long enough to generate unique PUK, IP address is not valid or subscription has more than one or has not any STB account whereas STB account code is not defined. Conflict is also raised when the same stb is assigned to another subscription STB account (exception detail contains subscription STB account code of that subscription), or subscription STB account has assigned another STB. Conflict is raised when STB serial number or MAC address exists or multiple STBs have same parameters.

1.3.15 Subscription – Create Enabled

This call combines operations `Create`, `Set Locality`, `Set Info` and `Enable`. It allows to create, set parameters and enable a new subscription and one STB account by one atomic operation.

Inputs: Data passed towards nangu.TV Platform

- Same data as for the `Create` operation.
- Same data as for the `Set Locality` operation.
- Same data as for the `Set Info` operation.
- Same data as for the `Enable` operation.
- Requested PUK length to use when no PUK is given.

Outputs: Possible data returned by nangu.TV Platform

- Used PUK value.
- NotFoundException – subscriber, subscription, STB account, locality, STB model, currency, tariff or country doesn't exist.
- ConflictException – the same subscription or STB account already exists, PUK was not given, the PUK length is not long enough to generate unique PUK, the PUK is not valid number, the same PUK value is already assigned, IP address is not valid or subscription has more than one or has not any STB account whereas STB account code is not defined. Conflict is also raised when the same stb is assigned to another subscription STB account (exception detail contains subscription STB account code of that subscription), or subscription STB account has assigned another STB. Conflict is raised when STB serial number or MAC address exists or multiple STBs have same parameters.

1.3.16 Subscription – Suspend

This call suspends an existing subscription. The service becomes not available for the subscription, but it can be re-activated. The service will not be billed for the suspended period. Subscription's state will be SUSPENDED.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Date of enabling the service.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription doesn't exist.
- ConflictException – subscription is not ready to be suspended or subscription is already suspended.

1.3.17 Subscription – Disable

This call disables a subscription. It is equal to subscription removal, but we cannot remove/delete a subscription immediately (we need to keep his/her identity for keeping history). This call is issued by provider's CRM after termination of a particular service contract with subscriber. Subscription's state will be REMOVED. All subscription's STB accounts are persistently removed.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription doesn't exist.

1.3.18 Subscription – Assign STB

This call explicitly assigns a STB to subscription's STB account. It is necessary when STB self-registration is not allowed. The subscription STB account has to be at least in state STB_SHIPPED to be able to use this method.

It is possible to reassign the same STB to just update STB attributes. The STB serial number or the STB MAC address has to equal to the old one to prevent unwanted overwriting by a different STB.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, one and only one subscription's STB account is used.
- STB model code.
- STB serial number.
- STB MAC address.
- STB note – any textual note.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription, STB account, or STB model doesn't exist
- ConflictException – request couldn't be fulfilled because the same stb is assigned to another subscription STB account, exception detail contains subscription STB account code of that subscription, or subscription STB account has assigned another STB. Conflict is raise when STB serial number or MAC address exists or multiple STBs have same parameters. The conflict is also raised when the subscription STB account is not enabled or subscription is disabled or has more than one or has not any STB account whereas STB account code is not defined.

1.3.19 Subscription – Unassign STB

This call unassigns STB from a subscription's STB account and changed his state to STB_SHIPPED. Useful when STB is broken. Note that the subscription is not automatically suspended when billing and STB account is active. It is needed to switch the subscription from state BILLING to an other state before usage of this service.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, one and only one subscription's STB account is used.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription or STB account doesn't exist
- ConflictException – STB account is in STB_ACTIVE state and doesn't have set PUK or subscription has more than one or has not any STB account whereas STB account code is not defined.

1.3.20 Subscription – Get STB Info

This call returns STB information of STB assigned with the given Subscription's STB account.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription STB account's (optional) identification code in provider's CRM. If code is not set, one and only one subscription's STB account is used.

Outputs: Possible data returned by nangu.TV Platform

- Stb information consisting of Stb model, serial number and MAC address.
- IP address assigned to Subscription's STB account
- NotFoundException – subscription or STB account doesn't exist
- ConflictException – Subscription has more than one or has not any STB account whereas STB account code is not defined.

1.3.21 Subscription – Create Subscription STB Account

This call adds a new subscription STB account to an existing subscription. STB account starts with state STB_ORDERED.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Subscription STB account's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription doesn't exist.
- ConflictException – request couldn't be fulfilled because same STB account exists.

1.3.22 Subscription – Set Subscription STB Account Info

This call is used for update subscription STB account's PUK and alowed IP adresses. Both fields can be set for newly created STB account or for update of an existing values by existing subscription STB account.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.
- Requested PUK (optional) - Setting an PUK to nil value will remove the stored PUK value.
- Allowed IP adresses (optional)

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription STB account or the country doesn't exist.
- ConflictException – The same PUK value is already assigned, the PUK is not valid number, or IP address is not valid.

1.3.23 Subscription – Get Subscription STB Account Info

This call returns complete information about subscription STB account.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- Subscription STB account details (all mandatory) – subcription STB account code, subscription code, subscriber code, subscription STB account state, create date.
- Subscription STB account details (all optional) – PUK, allowed IP addresses, IP address, assigned STB consisting of model code, serial number, MAC address and note.
- NotFoundException – Subscription STB account doesn't exist.

1.3.24 Subscription – Enable Subscription STB Account

This call enables activation of an existing subscription STB account. Subscription STB account's state will be switched to state STB_SHIPPED (subscription STB account in STB_ACTIVE state is not affected). If subscription is not in SUSPENDED state, the subscription STB account's state will be changed to STB_ACTIVE on the first STB use.

STB account is checked, whether meet the requirements of system config (E.g. valid PUK, filled allowed IPs, assigned STB, filled customer number). All possible required parameters can be set with this call for easier enabling of subscription.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.
- Puk (optional)
- Allowed IP adresses (optional)
- STB to assign (optional). Parameters are the same parameters as for Assign STB request.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription STB account or STB model doesn't exist.

- ConflictException – Subscription STB account is not ready to be enabled (reason of this is described in the exception) or is already enabled, the same PUK value is already assigned, PUK is not valid number or IP address is not valid. Conflict is also raised when the same stb is assigned to another subscription STB account (exception detail contains subscription STB account code), or subscription STB account has assigned another STB. Conflict is raised when STB serial number or MAC address exists or multiple STBs have same parameters.

1.3.25 Subscription – Enable Subscription STB Account with generated PUK

This call generates random PUK and enables activation of an existing subscription STB account. Subscription STB account enabling is the same as for Subscription – Enable Subscription STB Account.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.
- Allowed IP adresses (optional)
- STB to assign (optional). Parameters are the same parameters as for Assign STB request.
- Requested PUK length.

Outputs: Possible data returned by nangu.TV Platform

- Generated PUK value of given length. It is string with digits from 0 to 9 (e.g. 02975).
- NotFoundException – Subscription STB account or STB model doesn't exist.
- ConflictException – Subscription STB account is not ready to be enabled (reason of this is described in the exception), the PUK length is not long enough to generate unique PUK or IP address is not valid. Conflict is also raised when the same stb is assigned to another subscription STB account (exception detail contains subscription STB account code), or subscription STB account has assigned another STB. Conflict is raised when STB serial number or MAC address exists or multiple STBs have same parameters.

1.3.26 Subscription – Create Enabled Subscription STB Account

This call combines operations `Create Subscription STB Account`, `Set Subscription STB Account Info` and `Enable Subscription STB Account`. It allows to create, set PUK and enable a new subscription STB account by one atomic operation.

Inputs: Data passed towards nangu.TV Platform

- Same data as for the `Create Subscription STB Account` operation.
- Same data as for the `Set Subscription STB Account Info` operation.
- Same data as for the `Enable Subscription STB Account` operation.
- Requested PUK length to use when no PUK is given.

Outputs: Possible data returned by nangu·TV Platform

- Used PUK value.
- NotFoundException – Subscription or STB account doesn't exist.
- ConflictException – The same STB account already exists, PUK was not given, the same PUK value is already assigned, the PUK is not valid number, the PUK length is not long enough to generate unique PUK or IP address is not valid. Conflict is also raised when the same stb is assigned to another subscription STB account (exception detail contains subscription STB account code), or subscription STB account has assigned another STB. Conflict is raised when STB serial number or MAC address exists or multiple STBs have same parameters.

1.3.27 Subscription – Remove Subscription STB Account

This call persistently removes a subscription STB account.

This request is schedulable.

Inputs: Data passed towards nangu·TV Platform

- Subscription STB account's identification code in provider's CRM.

Outputs: Possible data returned by nangu·TV Platform

- NotFoundException – Subscription STB account doesn't exist.
- ConflictException – The conflict is also raised when subscription STB account is last STB_ACTIVE or STB_SHIPPED whereas subscription is in BILLING state.

1.3.28 Subscription – Get Subscription STB Accounts

This call returns list of existing Subscription's STB accounts and their states.

Inputs: Data passed towards nangu·TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu·TV Platform

- List of subscription STB account codes and their states.
- NotFoundException – Subscription doesn't exist.

1.3.29 Subscription – Change Tariff

This call sets new tariff for an existing subscription. If the new tariff contains tariff-mandatory offers not contained in the old tariff, such offers are automatically subscribed. If the new tariff doesn't contain offers contained in the old tariff, such offers are automatically unsubscribed,

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- New tariff code.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – subscription doesn't exist
- ConflictException – request couldn't be fulfilled, because new tarif doesn't have defined prices for all offers in subscription's currency.

Return Code: Errors are implemented as SOAP faults on the web service interface.

1.3.30 Subscription – Subscribe Offer

This call subscribes an offer for an existing subscription. This call replaces the former calls for subscription individual IPTV and SVOD packages. In case of an upgrading offer, all upgraded offers are automatically unsubscribed.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Offer code.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription or offer doesn't exist.
- ConflictException – request couldn't be fulfilled, because the Offer is not present in the subscription's tariff or subscription already has subscribed an offer, which upgrades this offer. In the latter case, the upgrade Offer should be unsubscribed first.

1.3.31 Subscription – Unsubscribe Offer

This call unsubscribes an offer for an existing subscription. This call replaces the former calls for unsubscribing individual IPTV and SVOD packages.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Offer code.
- A force flag – if set, unsubscribing is done even if the minimum subscription duration is not reached.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription or offer doesn't exist.
- ConflictException – request couldn't be fulfilled, because the offer is mandatory in subscription's current tariff or minimum subscription duration is not reached and the force flag is not set.

1.3.32 Subscription – Renew Offer

This call combines `unsubscribeOffer` and `subscribeOffer` together. Minimum subscription duration is ignored and unsubscribing is done even.

This request is schedulable.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Offer code.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription or offer doesn't exist.
- ConflictException – Same as for `unsubscribeOffer` and `subscribeOffer`.

1.3.33 Subscription – Get Offers

This call returns list of subscribed offers for given subscription.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- List of offer codes currently subscribed by the subscription.
- NotFoundException – Subscription doesn't exist.

1.3.34 Subscription – Enable TVoD

This call enables TVoD service for a subscription.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription doesn't exist.

1.3.35 Subscription – Disable TVoD

This call disables TVoD service for a subscription.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription doesn't exist.

1.3.36 Subscription – Get Charge Limit

This call returns the current value of subscription's charge limit. The charge limit limits subscription spending. It is decreased by every subscription charging (buying movie, offer fee, ...).

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- Charge limit.
- Subscription's currency.
- NotFoundException – Subscription doesn't exist.

1.3.37 Subscription – Set Charge Limit

This call allows to set subscription's charge limit. It could be called regularly to reset the limit.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Charge limit in subscription's currency.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription doesn't exist.

1.3.38 Subscription – Set Parental PIN

This call allows to set subscription's parental PIN.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- New parental PIN (4 digits) or empty string to reset the PIN.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription doesn't exist.
- ConflictException – The parental PIN is not valid number.

1.3.39 Subscription – Set Master PIN

This call allows to set subscription's master PIN. The master pin is used by users as purchase PIN when buying something.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- New master PIN (4 digits) or empty string to reset the PIN.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription doesn't exist.
- ConflictException – The master PIN is not valid number.

1.3.40 Subscription – Set Preferred Audio and Subtitles

This call allows to set subscription's preferred Audio and Subtitles language.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Language Code (or string "orig") for Audio.
- Language Code (or string "none"/"orig") for Subtitles.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription or language doesn't exist.

1.3.41 Subscription – Reset

This call resets all subscription's settings to their defaults.

Inputs: Data passed towards nangu.TV Platform

- List of subscription's identification codes in provider's CRM.
- New master (purchase) PIN (4 digits).
- New parental PIN (4 digits).

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – one of given subscriptions doesn't exist. No subscription is reset then.
- ConflictException – The master pin or the parental pin is not valid number.

1.3.42 Subscription – Send STB Message

This call puts a generic message into all subscription's STB accounts message queue. List of message types and their parameters is described bellow.

Inputs: Data passed towards nangu.TV Platform

- List of subscription's identification codes in provider's CRM.
- The message code, useful for message cancellation.
- Message type.
- Start time when to display the message.
- Message expiration after start time. Expired message is never shown.
- Attribute stbOn. Message with stbOn flag will be resend to STB startup on every restart until the message is not expired or canceled.
- List of message parameters and values.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – one of given subscriptions doesn't exist. No message is queued then.

AS message pool to JS communication

`REBOOT` – reboot the Set-top-box now.

`PORTAL_RELOAD` – reload the portal now.

`MSG_POPUP` – popup a flash message.

- `level` (enum = ONCE)
  - `ONCE` The message is displayed once when it is received.
  - `STB_ON` The message is to be displayed when it is received and then each time STB is switched on, until the message has expired.
  - `CHANNEL_SWITCH` The message is to be displayed when it is received and then after each channel switch, until the message has expired. The message overrides the previous CHANNEL_SWITCH message.
- `noHide` (boolean = FALSE) True if the user is not allowed to close the message.
- `title` (string) Title to be displayed.
- `text` (string) Text to be displayed.
- `duration` (number) Amount of time in seconds for how long should the text be displayed.
- `repetitionCounter` (number) Maximal count of message repetition.

`CHANNEL_SWITCH` – switch to the TV player and to the given channel.

- `channelKey` Channel to be displayed identified by channel key.

1.3.43 Subscription – Cancel Message

This call cancels a message from all subscription STB account's message queue, where subscription STB account is recipient of the message.

Inputs: Data passed towards nangu.TV Platform

- The message code that was used for Send Message.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – no such message exists.

1.3.44 Subscription – Cancel StbOn Messages

This call cancels all messages from all subscription's STB accounts message queue that have stbOn flag set to true.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – no such subscription exists.

1.3.45 Subscription – Send Subscription STB Account Message

This call puts a generic message into subscription STB accounts message queue. List of message types and their parameters is same as `sendStbMessage` described above.

Inputs: Data passed towards nangu.TV Platform

- List of subscription STB account's identification codes in provider's CRM.
- The message code, useful for message cancellation.
- Message type.
- Start time when to display the message.
- Message expiration after start time. Expired message is never shown.
- Attribute stbOn. Message with stbOn flag will be resend to STB startup on every restart until the message is not expired or cancelled.
- List of message parameters and values.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – one of given subscription STB accounts doesn't exist. No message is queued then.

1.3.46 Subscription – Cancel Subscription STB Account StbOn Messages

This call cancels all messages from all subscription STB account's message queue that have stbOn flag set to true.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – no such subscription STB account exists.

1.3.47 Subscription – Reboot STB

This call puts a message with STB reboot request into subscription STB accounts message queue.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – one of given subscription STB accounts doesn't exist. No message is queued then.

1.3.48 Subscription – Reboot All STB

This call puts a message with STB reboot request into all STB_ACTIVE subscription STB accounts message queue where subscription is in BILLING state.

Inputs: Data passed towards nangu.TV Platform

Outputs: Possible data returned by nangu.TV Platform

1.3.49 Subscription – Portal Reload

This call puts a message with portal reload request into subscription STB accounts message queue.

Inputs: Data passed towards nangu.TV Platform

- Subscription STB account's identification code in provider's CRM.

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – one of given subscription STB accounts doesn't exist. No message is queued then.

1.3.50 Subscription – Portal Reload All

This call puts a message with portal reload request into all STB_ACTIVE subscription STB accounts message queue where subscription is in BILLING state.

Inputs: Data passed towards nangu.TV Platform

Outputs: Possible data returned by nangu.TV Platform

1.3.51 Change subscription's portal version

This call changes the portal version of the given subscriptions.

Inputs: Data passed towards nangu.TV Platform

- Subscription's identification code in provider's CRM.
- Portal version. [PORTAL—SWS]

Outputs: Possible data returned by nangu.TV Platform

- NotFoundException – Subscription doesn't exist.

1.4 Self-Care Usage Verification – Server Interface

When Offers are subscribed through the customer self care, an external CRM system could verify the action.

1.4.1 Subscribe Offer in CRM

This call tells external CRM to subscribe an offer for an subscription. External system is expected to implement this contract:

Inputs: Data passed towards CRM System

- Subscription's identification code in provider's CRM.
- Offer code.
- ceil – flag whether subscribing was requested to be immediate (false), or since the next billing period (true).

Outputs: Possible data returned by CRM System

- subscribeDate – the date when the offer subscription becomes active.
- CrmNotFoundException – Subscription or offer doesn't exist.
- CrmConflictException – request couldn't be fulfilled, because a conflicting offer is already subscribed.

1.4.2 Unsubscribe Offer in CRM

This call tells external CRM to unsubscribe an offer for an subscription. External system is expected to implement this contract:

Inputs: Data passed towards CRM System

- Subscription's identification code in provider's CRM.
- Offer code.

Outputs: Possible data returned by CRM System

- unsubscribeDate – the date when the offer subscription is unsubscribed.
- CrmNotFoundException – Subscription or offer doesn't exist.
- CrmConflictException – request couldn't be fulfilled, because a conflicting offer is already subscribed.

1.5 Request scheduling

Some of provisioning requests are scheduleable, i.e. it can be specified, when the request should be executed. Schedulable requests are marked in the interface specification. They accept additional parameters `dueDate` and `scheduleOption`. If `dueDate` is not specified, the request is executed immediately. The `scheduleOption` atribute can have one of the following values

INSERT – the action is inserted into the schedule, actions regarding the same object are kept in the schedule.

FIRST – the action is inserted into the schedule, actions regarding the same object, that are planned before the added action, are cancelled.

LAST – the action is inserted into the schedule, actions regarding the same object, that are planned after the added action, are cancelled.

ONLY – the action is inserted into the schedule, all currently planned actions regarding the same object are cancelled.

Note: request scheduling is not yet implemented. Its addition will be backward compatible because scheduling attributes are optional.

1.6 Web Services – SOAP

Web services invoked through SOAP share a common pattern – values of successful call are returned as output parameters, while values of unsuccessful calls are returned as SOAP faults.

1.6.1 WSDL for Subscriber services

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://itonis.net/tve/schema/admin
   /provisioning/subscriber/ns" xmlns:tns="http://itonis.net/tve/
   schema/admin/provisioning/subscriber/ns" xmlns:wsdlsoap="http://
   schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://www.w3.org
   /2003/05/soap-envelope" xmlns:ns1="http://itonis.net/tve/schema/
   admin/subscriber" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
   xmlns:ns2="http://provisioning.jws.admin.web.tve.itonis.net"
   xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="
   http://itonis.net/tve/schema/common/provisioning"
   elementFormDefault="unqualified" targetNamespace="http://itonis.
   net/tve/schema/common/provisioning">
    <xsd:annotation>
        <xsd:documentation>Common provisioning datatypes. Version: $
           Revision: 56717 $ $Date: 2010-01-19 20:50:54 +0100 (Tue,
           19 Jan 2010) $</xsd:documentation>
    </xsd:annotation>

    <xsd:simpleType name="Code">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="32"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Puk">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="32"/>
            <xsd:pattern value="[0-9]{1,32}|([0-9a-f]{2}:){5}[0-9a-f
               ]{2}|"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Pin">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9]{4}|"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="SubscriptionState">
        <xsd:restriction base="xsd:string">
```

```
            <xsd:enumeration value="NEW"/>
            <xsd:enumeration value="BILLING"/>
            <xsd:enumeration value="SUSPENDED"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="SubscriptionStbAccountState">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STB_ORDERED"/>
            <xsd:enumeration value="STB_SHIPPED"/>
            <xsd:enumeration value="STB_ACTIVE"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Name">
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="64"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Email">
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255"/>
            <xsd:pattern value="[a-zA-Z0-9\._%\+\-]+@([a-zA-Z0 ↩
               -9\-]+\.)+[a-zA-Z]{2,4}|"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Audience">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="NOT_RATED"/>
            <xsd:enumeration value="OVER_12"/>
            <xsd:enumeration value="OVER_15"/>
            <xsd:enumeration value="OVER_18"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="MessageType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="MSG_POPUP"/>
            <xsd:enumeration value="CHANNEL_SWITCH"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Currency">
        <xsd:restriction base="xsd:string">
            <xsd:length value="3"/>
        </xsd:restriction>
    </xsd:simpleType>
```

```
<xsd:complexType name="Address">
    <xsd:sequence>
        <xsd:element minOccurs="0" name="city" type="xsd:string"/ ↩
           >
        <xsd:element minOccurs="0" name="street" type="xsd:string ↩
           "/>
        <xsd:element minOccurs="0" name="county" type="xsd:string ↩
           "/>
        <xsd:element minOccurs="0" name="countryIso2">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:length value="2"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element minOccurs="0" name="zipCode" type=" ↩
           xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Stb">
    <xsd:annotation>
        <xsd:documentation>Set-Top-Box.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="modelCode" type="tns:Code"/>
        <xsd:element name="serialNumber">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:minLength value="1"/>
                    <xsd:maxLength value="50"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="macAddress">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:minLength value="17"/>
                    <xsd:maxLength value="17"/>
                    <xsd:pattern value="([0-9a-f]{2}:){5}[0-9a-f ↩
                       ]{2}|"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element minOccurs="0" name="note" type="xsd:string"/ ↩
           >
    </xsd:sequence>
</xsd:complexType>
```

```
    <xsd:simpleType name="PortalVersion">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="PORTAL"/>
            <xsd:enumeration value="SWS"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns=" ↩
   http://itonis.net/tve/schema/common" elementFormDefault=" ↩
   unqualified" targetNamespace="http://itonis.net/tve/schema/common" ↩
   >
    <xsd:annotation>
        <xsd:documentation>Common datatypes. Version: $Revision:  ↩
           33913 $ $Date: 2007-09-17 10:36:48 +0200 (Mon, 17 Sep  ↩
           2007) $</xsd:documentation>
    </xsd:annotation>

    <xsd:simpleType name="Date">
        <xsd:restriction base="xsd:date">
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="DateTime">
        <xsd:restriction base="xsd:dateTime">
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="SchedulableRequest">
        <xsd:sequence>
            <xsd:element minOccurs="0" name="dueDate" type=" ↩
               tns:DateTime"/>
            <xsd:element minOccurs="0" name="scheduleOption">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="INSERT"/>
                        <xsd:enumeration value="FIRST"/>
                        <xsd:enumeration value="LAST"/>
                        <xsd:enumeration value="ONLY"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:common ↩
   ="http://itonis.net/tve/schema/common" xmlns:provisioning="http:// ↩
   itonis.net/tve/schema/common/provisioning" xmlns:tns="http:// ↩
```

```
itonis.net/tve/schema/admin/subscriber" elementFormDefault=" ↩
unqualified" targetNamespace="http://itonis.net/tve/schema/admin/ ↩
subscriber">

 <xsd:import namespace="http://itonis.net/tve/schema/common"/>
 <xsd:import namespace="http://itonis.net/tve/schema/common/ ↩
    provisioning"/>

 <xsd:annotation>
     <xsd:documentation>Subscriber provisioning interface.  ↩
        Version: $Revision: 56717 $ $Date: 2010-01-19 20:50:54  ↩
        +0100 (Tue, 19 Jan 2010) $</xsd:documentation>
 </xsd:annotation>

 <xsd:element name="createRequest">
     <xsd:annotation>
         <xsd:documentation>Always creates new subscriber.</ ↩
            xsd:documentation>
     </xsd:annotation>
     <xsd:complexType>
         <xsd:complexContent>
             <xsd:extension base="common:SchedulableRequest">
                  <xsd:sequence>
                     <xsd:element name="subscriberCode" type=" ↩
                        provisioning:Code"/>
                     <xsd:element minOccurs="0" name=" ↩
                        customerNumber" type="provisioning:Code"/>
                 </xsd:sequence>
             </xsd:extension>
         </xsd:complexContent>
     </xsd:complexType>
 </xsd:element>

 <xsd:element name="disableRequest">
     <xsd:annotation>
         <xsd:documentation>Disables subscriber.</ ↩
            xsd:documentation>
     </xsd:annotation>
     <xsd:complexType>
         <xsd:complexContent>
             <xsd:extension base="common:SchedulableRequest">
                 <xsd:sequence>
                     <xsd:element name="subscriberCode" type=" ↩
                        provisioning:Code"/>
                 </xsd:sequence>
             </xsd:extension>
         </xsd:complexContent>
     </xsd:complexType>
 </xsd:element>
```

```
<xsd:element name="setInfoRequest">
    <xsd:annotation>
        <xsd:documentation>Updates information about subscriber.  ↩
           Only provided properties are updated.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriberCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name="customerNumber" type ↩
               ="provisioning:Code"/>
            <xsd:element minOccurs="0" name="firstName" type=" ↩
               provisioning:Name"/>
            <xsd:element minOccurs="0" name="middleName" type=" ↩
               provisioning:Name"/>
            <xsd:element minOccurs="0" name="lastName" type=" ↩
               provisioning:Name"/>
            <xsd:element minOccurs="0" name="sex" type="tns:Sex"/ ↩
               >
            <xsd:element minOccurs="0" name="email" type=" ↩
               provisioning:Email"/>
            <xsd:element minOccurs="0" name="address" nillable=" ↩
               true" type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="note" type=" ↩
               xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getInfoRequest">
    <xsd:annotation>
        <xsd:documentation>Returns information about subscriber.< ↩
           /xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriberCode" type=" ↩
               provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getInfoResponse">
    <xsd:annotation>
        <xsd:documentation>Information about subscriber.</ ↩
           xsd:documentation>
    </xsd:annotation>
```

```
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriberCode" type=" ↩
                    provisioning:Code"/>
                <xsd:element minOccurs="0" name="customerNumber" type ↩
                    ="provisioning:Code"/>
                <xsd:element minOccurs="0" name="firstName" type=" ↩
                    provisioning:Name"/>
                <xsd:element minOccurs="0" name="middleName" type=" ↩
                    provisioning:Name"/>
                <xsd:element minOccurs="0" name="lastName" type=" ↩
                    provisioning:Name"/>
                <xsd:element minOccurs="0" name="sex" type="tns:Sex"/ ↩
                    >
                <xsd:element minOccurs="0" name="username" type=" ↩
                    provisioning:Code"/>
                <xsd:element minOccurs="0" name="email" type=" ↩
                    provisioning:Email"/>
                <xsd:element minOccurs="0" name="address" type=" ↩
                    provisioning:Address"/>
                <xsd:element minOccurs="0" name="note" type=" ↩
                    xsd:string"/>
                <xsd:element name="createDate" type="common:DateTime" ↩
                    />
                <xsd:element ref="tns:getSubscriptionsResponse"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="getSubscriptionsRequest">
        <xsd:annotation>
            <xsd:documentation>Returns list of subscriber's  ↩
                subscriptions.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriberCode" type=" ↩
                    provisioning:Code"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="getSubscriptionsResponse">
        <xsd:annotation>
            <xsd:documentation>Lists subscriber's subscriptions and  ↩
                their states.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" minOccurs="0" name ↩
               ="subscriptions">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="subscriptionCode" type ↩
                           ="provisioning:Code"/>
                        <xsd:element name="subscriptionState"  ↩
                           type="provisioning:SubscriptionState"/ ↩
                           >
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="searchRequest">
    <xsd:annotation>
        <xsd:documentation>Returns list of subscribers passed to  ↩
           search parameters.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element minOccurs="0" name="searchText" type=" ↩
               xsd:string"/>
            <xsd:element minOccurs="0" name="searchType" type=" ↩
               tns:SubscriberSearchType"/>
            <xsd:element minOccurs="0" name="subscriptionState"  ↩
               type="provisioning:SubscriptionState"/>
            <xsd:element minOccurs="0" name=" ↩
               subscriptionStbAccountState" type=" ↩
               provisioning:SubscriptionStbAccountState"/>
            <xsd:element minOccurs="0" name="localityCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name="offerCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name="firstResult" type=" ↩
               xsd:int"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="searchResponse">
    <xsd:annotation>
        <xsd:documentation>Lists subscribers.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="count" type="xsd:int"/>
```

```
                <xsd:element maxOccurs="100" minOccurs="0" name=" ↩
                    subscribers">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="subscriberCode" type=" ↩
                                provisioning:Code"/>
                            <xsd:element name="subscriberFullName"  ↩
                                type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="Sex">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="MALE"/>
            <xsd:enumeration value="FEMALE"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="SubscriberSearchType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="SUBSCRIBER_CODE"/>
            <xsd:enumeration value="SUBSCRIBER_NAME"/>
            <xsd:enumeration value="USERNAME"/>
            <xsd:enumeration value="CUSTOMER_NUMBER"/>
            <xsd:enumeration value="STB_SERIAL"/>
            <xsd:enumeration value="IP_ADDRESS"/>
            <xsd:enumeration value="MAC_ADDRESS"/>
            <xsd:enumeration value="SUBSCRIPTION_CODE"/>
            <xsd:enumeration value="STB_DN"/>
            <xsd:enumeration value="SUBSCRIPTION_STB_ACCOUNT_CODE"/>
            <xsd:enumeration value="PUK"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"  ↩
   attributeFormDefault="qualified" elementFormDefault="qualified"  ↩
   targetNamespace="http://provisioning.jws.admin.web.tve.itonis.net" ↩
   >
<xsd:complexType name="NotFoundException"/>
<xsd:complexType name="ConflictException"/>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"  ↩
   attributeFormDefault="qualified" elementFormDefault="qualified"  ↩
   targetNamespace="http://itonis.net/tve/schema/admin/provisioning/ ↩
```

```
   subscriber/ns">
<xsd:element name="NotFoundException" type="ns2:NotFoundException"/>
<xsd:element name="ConflictException" type="ns2:ConflictException"/>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="disableRequest">
    <wsdl:part name="disableRequest" element="ns1:disableRequest"/>
  </wsdl:message>
  <wsdl:message name="createResponse">
  </wsdl:message>
  <wsdl:message name="searchRequest">
    <wsdl:part name="searchRequest" element="ns1:searchRequest"/>
  </wsdl:message>
  <wsdl:message name="disableResponse">
  </wsdl:message>
  <wsdl:message name="getSubscriptionsRequest">
    <wsdl:part name="getSubscriptionsRequest" element=" ↩
       ns1:getSubscriptionsRequest"/>
  </wsdl:message>
  <wsdl:message name="ConflictException">
    <wsdl:part name="ConflictException" element=" ↩
       tns:ConflictException"/>
  </wsdl:message>
  <wsdl:message name="setInfoResponse">
  </wsdl:message>
  <wsdl:message name="NotFoundException">
    <wsdl:part name="NotFoundException" element=" ↩
       tns:NotFoundException"/>
  </wsdl:message>
  <wsdl:message name="setInfoRequest">
    <wsdl:part name="setInfoRequest" element="ns1:setInfoRequest"/>
  </wsdl:message>
  <wsdl:message name="getInfoResponse">
    <wsdl:part name="getInfoResponse" element="ns1:getInfoResponse"/>
  </wsdl:message>
  <wsdl:message name="searchResponse">
    <wsdl:part name="searchResponse" element="ns1:searchResponse"/>
  </wsdl:message>
  <wsdl:message name="getInfoRequest">
    <wsdl:part name="getInfoRequest" element="ns1:getInfoRequest"/>
  </wsdl:message>
  <wsdl:message name="getSubscriptionsResponse">
    <wsdl:part name="getSubscriptionsResponse" element=" ↩
       ns1:getSubscriptionsResponse"/>
  </wsdl:message>
  <wsdl:message name="createRequest">
    <wsdl:part name="createRequest" element="ns1:createRequest"/>
  </wsdl:message>
  <wsdl:portType name="SubscriberEndpointPortType">
```

```
    <wsdl:operation name="search">
      <wsdl:input name="searchRequest" message="tns:searchRequest"/>
      <wsdl:output name="searchResponse" message="tns:searchResponse" ↩
         />
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="getSubscriptions">
      <wsdl:input name="getSubscriptionsRequest" message=" ↩
         tns:getSubscriptionsRequest"/>
      <wsdl:output name="getSubscriptionsResponse" message=" ↩
         tns:getSubscriptionsResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="getInfo">
      <wsdl:input name="getInfoRequest" message="tns:getInfoRequest"/ ↩
         >
      <wsdl:output name="getInfoResponse" message=" ↩
         tns:getInfoResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="create">
      <wsdl:input name="createRequest" message="tns:createRequest"/>
      <wsdl:output name="createResponse" message="tns:createResponse" ↩
         />
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="setInfo">
      <wsdl:input name="setInfoRequest" message="tns:setInfoRequest"/ ↩
         >
      <wsdl:output name="setInfoResponse" message=" ↩
         tns:setInfoResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="disable">
      <wsdl:input name="disableRequest" message="tns:disableRequest"/ ↩
         >
      <wsdl:output name="disableResponse" message=" ↩
         tns:disableResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
```

```
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SubscriberEndpointHttpBinding" type=" ↩
      tns:SubscriberEndpointPortType">
    <wsdlsoap:binding style="document" transport="http://schemas. ↩
        xmlsoap.org/soap/http"/>
    <wsdl:operation name="search">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="searchRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="searchResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getSubscriptions">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSubscriptionsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSubscriptionsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getInfoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getInfoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="create">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="createRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createResponse">
        <wsdlsoap:body use="literal"/>
```

```
      </wsdl:output>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setInfoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setInfoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="disable">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="disableRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="disableResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SubscriberEndpoint">
    <wsdl:port name="SubscriberEndpointHttpPort" binding=" ↩
       tns:SubscriberEndpointHttpBinding">
      <wsdlsoap:address location="http://localhost/services/ ↩
         SubscriberEndpoint"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

1.6.2 WSDL for Subscription services

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions targetNamespace="http://itonis.net/tve/schema/admin
   /provisioning/subscription/ns" xmlns:tns="http://itonis.net/tve/
   schema/admin/provisioning/subscription/ns" xmlns:wsdlsoap="http://
   schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://www.w3.org
   /2003/05/soap-envelope" xmlns:ns1="http://itonis.net/tve/schema/
   admin/subscription" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
   xmlns:ns2="http://provisioning.jws.admin.web.tve.itonis.net"
   xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="
   http://itonis.net/tve/schema/common/provisioning"
   elementFormDefault="unqualified" targetNamespace="http://itonis.
   net/tve/schema/common/provisioning">
    <xsd:annotation>
        <xsd:documentation>Common provisioning datatypes. Version: $
           Revision: 56717 $ $Date: 2010-01-19 20:50:54 +0100 (Tue,
           19 Jan 2010) $</xsd:documentation>
    </xsd:annotation>

    <xsd:simpleType name="Code">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="32"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Puk">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="32"/>
            <xsd:pattern value="[0-9]{1,32}|([0-9a-f]{2}:){5}[0-9a-f
               ]{2}|"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Pin">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9]{4}|"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="SubscriptionState">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="NEW"/>
            <xsd:enumeration value="BILLING"/>
            <xsd:enumeration value="SUSPENDED"/>
```

```
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="SubscriptionStbAccountState">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STB_ORDERED"/>
            <xsd:enumeration value="STB_SHIPPED"/>
            <xsd:enumeration value="STB_ACTIVE"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Name">
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="64"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Email">
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255"/>
            <xsd:pattern value="[a-zA-Z0-9\._%\+\-]+@([a-zA-Z0 ↩
               -9\-]+\.)+[a-zA-Z]{2,4}|"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Audience">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="NOT_RATED"/>
            <xsd:enumeration value="OVER_12"/>
            <xsd:enumeration value="OVER_15"/>
            <xsd:enumeration value="OVER_18"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="MessageType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="MSG_POPUP"/>
            <xsd:enumeration value="CHANNEL_SWITCH"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Currency">
        <xsd:restriction base="xsd:string">
            <xsd:length value="3"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="Address">
        <xsd:sequence>
```

```
            <xsd:element minOccurs="0" name="city" type="xsd:string"/>
            <xsd:element minOccurs="0" name="street" type="xsd:string"/>
            <xsd:element minOccurs="0" name="county" type="xsd:string"/>
            <xsd:element minOccurs="0" name="countryIso2">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:length value="2"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element minOccurs="0" name="zipCode" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Stb">
        <xsd:annotation>
            <xsd:documentation>Set-Top-Box.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="modelCode" type="tns:Code"/>
            <xsd:element name="serialNumber">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="1"/>
                        <xsd:maxLength value="50"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="macAddress">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="17"/>
                        <xsd:maxLength value="17"/>
                        <xsd:pattern value="([0-9a-f]{2}:){5}[0-9a-f]{2}|"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element minOccurs="0" name="note" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="PortalVersion">
        <xsd:restriction base="xsd:string">
```

```
            <xsd:enumeration value="PORTAL"/>
            <xsd:enumeration value="SWS"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns=" ↩
   http://itonis.net/tve/schema/common" elementFormDefault=" ↩
   unqualified" targetNamespace="http://itonis.net/tve/schema/common" ↩
   >
    <xsd:annotation>
        <xsd:documentation>Common datatypes. Version: $Revision:  ↩
           33913 $ $Date: 2007-09-17 10:36:48 +0200 (Mon, 17 Sep  ↩
           2007) $</xsd:documentation>
    </xsd:annotation>

    <xsd:simpleType name="Date">
        <xsd:restriction base="xsd:date">
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="DateTime">
        <xsd:restriction base="xsd:dateTime">
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="SchedulableRequest">
        <xsd:sequence>
            <xsd:element minOccurs="0" name="dueDate" type=" ↩
               tns:DateTime"/>
            <xsd:element minOccurs="0" name="scheduleOption">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="INSERT"/>
                        <xsd:enumeration value="FIRST"/>
                        <xsd:enumeration value="LAST"/>
                        <xsd:enumeration value="ONLY"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:common ↩
   ="http://itonis.net/tve/schema/common" xmlns:provisioning="http:// ↩
   itonis.net/tve/schema/common/provisioning" xmlns:tns="http:// ↩
   itonis.net/tve/schema/admin/subscription" elementFormDefault=" ↩
   unqualified" targetNamespace="http://itonis.net/tve/schema/admin/ ↩
   subscription">
```

```
<xsd:import namespace="http://itonis.net/tve/schema/common"/>
<xsd:import namespace="http://itonis.net/tve/schema/common/ ↩
   provisioning"/>

<xsd:annotation>
    <xsd:documentation>Subscription provisioning interface.  ↩
       Version: $Revision: 56708 $ $Date: 2010-01-19 16:42:44  ↩
       +0100 (Tue, 19 Jan 2010) $</xsd:documentation>
</xsd:annotation>

<xsd:element name="createRequest">
    <xsd:annotation>
        <xsd:documentation>Creates new subscription with one stb  ↩
           account for given subscriber.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="common:SchedulableRequest">
                <xsd:sequence>
                    <xsd:element name="subscriberCode" type=" ↩
                       provisioning:Code"/>
                    <xsd:element name="subscriptionCode" type=" ↩
                       provisioning:Code"/>
                    <xsd:element minOccurs="0" name=" ↩
                       subscriptionStbAccountCode" type=" ↩
                       provisioning:Code">
                        <xsd:annotation>
                            <xsd:documentation>When no  ↩
                               subscriptionStbAccountCode is  ↩
                               given, a subscriptionCode is used  ↩
                               instead.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="currencyCode" type=" ↩
                       provisioning:Code"/>
                    <xsd:element minOccurs="0" name=" ↩
                       altCurrencyCode" type="provisioning:Code"/ ↩
                       >
                    <xsd:element name="tariffCode" type=" ↩
                       provisioning:Code"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="createWithoutStbAccountRequest">
    <xsd:annotation>
```

```
            <xsd:documentation>Creates new subscription without any
               stb account for given subscriber.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="common:SchedulableRequest">
                    <xsd:sequence>
                        <xsd:element name="subscriberCode" type="
                           provisioning:Code"/>
                        <xsd:element name="subscriptionCode" type="
                           provisioning:Code"/>
                        <xsd:element name="currencyCode" type="
                           provisioning:Code"/>
                        <xsd:element minOccurs="0" name="
                           altCurrencyCode" type="provisioning:Code"/
                           >
                        <xsd:element name="tariffCode" type="
                           provisioning:Code"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="setLocalityRequest">
        <xsd:annotation>
            <xsd:documentation>Sets provisioning locality for the
               subscription.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="common:SchedulableRequest">
                     <xsd:sequence>
                         <xsd:element name="subscriptionCode" type="
                            provisioning:Code"/>
                         <xsd:element name="localityCode" type="
                            provisioning:Code"/>
                     </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="enableRequest">
        <xsd:annotation>
            <xsd:documentation>Enables activation of an existing
               subscription.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
```

```
            <xsd:complexContent>
                <xsd:extension base="common:SchedulableRequest">
                    <xsd:sequence>
                        <xsd:element name="subscriptionCode" type=" ↩
                            provisioning:Code"/>
                        <xsd:element minOccurs="0" name=" ↩
                            subscriptionStbAccountCode" type=" ↩
                            provisioning:Code"/>
                        <xsd:element minOccurs="0" name="enableDate"  ↩
                            type="common:Date">
                            <xsd:annotation>
                                <xsd:documentation>Service billing is ↩
                                     started automatically on this  ↩
                                    date. When no date is given and  ↩
                                    subscription is NEW, the billing  ↩
                                    is started on first STB usage.</ ↩
                                    xsd:documentation>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element minOccurs="0" name="puk" type=" ↩
                            provisioning:Puk"/>
                        <xsd:element minOccurs="0" name="allowedIps"  ↩
                            type="xsd:string">
                            <xsd:annotation>
                                <xsd:documentation>List of allowed  ↩
                                    IPs when SystemConfig.anyIpAllowed ↩
                                     is disabled. Multiple IP  ↩
                                    addresses could be separated by  ↩
                                    space, comma or semicolon.</ ↩
                                    xsd:documentation>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element minOccurs="0" name="stb" type=" ↩
                            provisioning:Stb"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="enableWithGeneratedPukRequest">
        <xsd:annotation>
            <xsd:documentation>Generates PUK and enables activation  ↩
                of an existing subscription. Generated PUK is returned ↩
                 in response.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="tns:enableRequest"/>
```

```
            <xsd:element name="pukLength">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:int">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="255"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="enableWithGeneratedPukResponse">
    <xsd:annotation>
        <xsd:documentation>Holds generated PUK.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="puk" type="provisioning:Puk"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="createEnabledRequest">
    <xsd:annotation>
        <xsd:documentation>Creates enabled subscription ready to  ↩
           be ativated. Combines create(), setInfo() and enable() ↩
            into one operation.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:createRequest"/>
            <xsd:element minOccurs="0" name="localityCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name="provisioningAddress" ↩
                type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="shippingAddress"  ↩
               type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="note" type=" ↩
               xsd:string"/>
            <xsd:element minOccurs="0" name="allowedIps" type=" ↩
               xsd:string">
                <xsd:annotation>
                    <xsd:documentation>List of allowed IPs when  ↩
                       SystemConfig.anyIpAllowed is disabled.  ↩
                       Multiple IP addresses could be separated  ↩
                       by space, comma or semicolon.</ ↩
                       xsd:documentation>
```

```
                </xsd:annotation>
            </xsd:element>
            <xsd:element minOccurs="0" name="stb" type=" ↩
               provisioning:Stb"/>
            <xsd:element minOccurs="0" name="enableDate" type=" ↩
               common:Date"/>
            <xsd:element minOccurs="0" name="puk" type=" ↩
               provisioning:Puk"/>
            <xsd:element default="8" minOccurs="0" name=" ↩
               pukLength">
                <xsd:annotation>
                    <xsd:documentation>When no PUK is given, a  ↩
                       PUK of this length is generated.</ ↩
                       xsd:documentation>
                </xsd:annotation>
                <xsd:simpleType>
                    <xsd:restriction base="xsd:int">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="255"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="createEnabledResponse">
    <xsd:annotation>
        <xsd:documentation>Holds the used PUK.</xsd:documentation ↩
           >
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="puk" type="provisioning:Puk"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="suspendRequest">
    <xsd:annotation>
        <xsd:documentation>Suspends an existing subscription. The ↩
            service will not be available, but can be reactivated ↩
           . The service will not be billed for suspended period. ↩
           </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="common:SchedulableRequest">
                <xsd:sequence>
```

```
                    <xsd:element name="subscriptionCode" type=" ↩
                        provisioning:Code"/>
                    <xsd:element minOccurs="0" name="enableDate"  ↩
                        type="common:Date">
                        <xsd:annotation>
                            <xsd:documentation>When date is given ↩
                                , service is automatically enabled ↩
                                 on this date.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="disableRequest">
    <xsd:annotation>
        <xsd:documentation>Disables a subscription.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="common:SchedulableRequest">
                <xsd:sequence>
                    <xsd:element name="subscriptionCode" type=" ↩
                        provisioning:Code"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="setInfoRequest">
    <xsd:annotation>
        <xsd:documentation>Updates additional information about  ↩
            subscription.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
            <xsd:element minOccurs="0" name=" ↩
                subscriptionStbAccountCode" type=" ↩
                provisioning:Code"/>
            <xsd:element minOccurs="0" name="puk" nillable="true" ↩
                type="provisioning:Puk">
                <xsd:annotation>
```

```
                    <xsd:documentation>Personal Unblocking Key ↩
                        that the customer has to use on ↩
                        registration and other actions.</ ↩
                        xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element minOccurs="0" name="provisioningAddress" ↩
                nillable="true" type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="shippingAddress" ↩
                nillable="true" type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="note" type=" ↩
                xsd:string"/>
            <xsd:element minOccurs="0" name="allowedIps" type=" ↩
                xsd:string">
                <xsd:annotation>
                    <xsd:documentation>List of allowed IPs when ↩
                        SystemConfig.anyIpAllowed is disabled. ↩
                        Multiple IP addresses could be separated ↩
                        by space, comma or semicolon.</ ↩
                        xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getInfoRequest">
    <xsd:annotation>
        <xsd:documentation>Returns information about subscription ↩
            .</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getInfoResponse">
    <xsd:annotation>
        <xsd:documentation>Information about requested ↩
            subscription.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
```

```
            <xsd:element name="subscriberCode" type=" ↩
               provisioning:Code"/>
            <xsd:element name="currencyCode" type=" ↩
               provisioning:Currency"/>
            <xsd:element minOccurs="0" name="altCurrencyCode"  ↩
               type="provisioning:Currency"/>
            <xsd:element name="tariffCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name="localityCode" type=" ↩
               provisioning:Code"/>
            <xsd:element name="subscriptionState" type=" ↩
               provisioning:SubscriptionState"/>
            <xsd:element minOccurs="0" name="enableDate" type=" ↩
               common:Date"/>
            <xsd:element minOccurs="0" name="parentalPin" type=" ↩
               provisioning:Pin"/>
            <xsd:element minOccurs="0" name="chargeLimit" type=" ↩
               xsd:double"/>
            <xsd:element name="tvodOrdered" type="xsd:boolean"/>
            <xsd:element name="parentalSearch" type=" ↩
               provisioning:Audience"/>
            <xsd:element name="parentalDetail" type=" ↩
               provisioning:Audience"/>
            <xsd:element minOccurs="0" name="shippingAddress"  ↩
               type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="provisioningAddress" ↩
                type="provisioning:Address"/>
            <xsd:element minOccurs="0" name="note" type=" ↩
               xsd:string"/>
            <xsd:element name="createDate" type="common:DateTime" ↩
               />
            <xsd:element name="npvrSecondsUsed" type="xsd:int"/>
            <xsd:element name="npvrSecondsMax" type="xsd:int"/>
            <xsd:element name="npvrHistoryDays" type="xsd:int"/>
            <xsd:element ref="tns:getOffersResponse"/>
            <xsd:element ref=" ↩
               tns:getSubscriptionStbAccountsResponse"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="assignStbRequest">
    <xsd:annotation>
        <xsd:documentation>Explicitly assigns a STB to a  ↩
           subscription stb account.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name=" ↩
               subscriptionStbAccountCode" type=" ↩
               provisioning:Code"/>
            <xsd:element name="stb" type="provisioning:Stb"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="unassignStbRequest">
    <xsd:annotation>
        <xsd:documentation>Unassigns STB from a subscription.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name=" ↩
               subscriptionStbAccountCode" type=" ↩
               provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getStbInfoRequest">
    <xsd:annotation>
        <xsd:documentation>Returns STB information assigned with  ↩
           the given subscription.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
            <xsd:element minOccurs="0" name=" ↩
               subscriptionStbAccountCode" type=" ↩
               provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getStbInfoResponse">
    <xsd:annotation>
        <xsd:documentation>Contains STB information when there is ↩
            one.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element minOccurs="0" name="stb" type=" ↩
               provisioning:Stb"/>
            <xsd:element minOccurs="0" name="ipAddress" type=" ↩
               xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="changeTariffRequest">
    <xsd:annotation>
        <xsd:documentation>Sets new tariffs for an existing  ↩
           subscription. Mandatory packages are automatically  ↩
           subscribed. Noneligible packages are automatically  ↩
           unsubscribed.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="common:SchedulableRequest">
                <xsd:sequence>
                    <xsd:element name="subscriptionCode" type=" ↩
                       provisioning:Code"/>
                    <xsd:element name="tariffCode" type=" ↩
                       provisioning:Code"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="enableTvodRequest">
    <xsd:annotation>
        <xsd:documentation>Enables TVoD service for a  ↩
           subscription.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="disableTvodRequest">
    <xsd:annotation>
        <xsd:documentation>Disables TVoD service for a  ↩
           subscription.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="subscribeOfferRequest">
    <xsd:annotation>
        <xsd:documentation>Subscribes offer for subscription.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
            <xsd:element name="offerCode" type="provisioning:Code ↩
               "/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="renewOfferRequest">
    <xsd:annotation>
        <xsd:documentation>Renews offer for subscription.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
               provisioning:Code"/>
            <xsd:element name="offerCode" type="provisioning:Code ↩
               "/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="unsubscribeOfferRequest">
    <xsd:annotation>
        <xsd:documentation>Unsubscribes offer for subscription.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="common:SchedulableRequest">
                <xsd:sequence>
                    <xsd:element name="subscriptionCode" type=" ↩
                       provisioning:Code"/>
                    <xsd:element name="offerCode" type=" ↩
                       provisioning:Code"/>
```

```
                    <xsd:element name="force" type="xsd:boolean">
                        <xsd:annotation>
                            <xsd:documentation>Whether to ignore ↩
                                minimal subscription duration.</ ↩
                                xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getOffersRequest">
    <xsd:annotation>
        <xsd:documentation>Gets list of subscribed offers.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getOffersResponse">
    <xsd:annotation>
        <xsd:documentation>Lists codes of subscribed offers.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name ↩
                ="offerCodes" type="provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="setChargeLimitRequest">
    <xsd:annotation>
        <xsd:documentation>Sets subscription's charge limit.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
            <xsd:element name="chargeLimit" type="xsd:double">
```

```
                <xsd:annotation>
                    <xsd:documentation>Limit in subscription's ↩
                        currency.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getChargeLimitRequest">
    <xsd:annotation>
        <xsd:documentation>Returns subscription's charge limit.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getChargeLimitResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="chargeLimit" type="xsd:double">
                <xsd:annotation>
                    <xsd:documentation>Limit in subscription's ↩
                        currency.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="currencyCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="setParentalPinRequest">
    <xsd:annotation>
        <xsd:documentation>Sets subscription's parent PIN.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
            <xsd:element name="parentalPin" type=" ↩
                provisioning:Pin"/>
        </xsd:sequence>
```

```
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="setMasterPinRequest">
        <xsd:annotation>
            <xsd:documentation>Sets subscription's master PIN.</ ↩
               xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionCode" type=" ↩
                   provisioning:Code"/>
                <xsd:element name="masterPin" type="provisioning:Pin" ↩
                   />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="setPreferredAudioSubtitlesRequest">
        <xsd:annotation>
            <xsd:documentation>Sets subscription's preferred audio  ↩
               and subtitles language.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionCode" type=" ↩
                   provisioning:Code"/>
                <xsd:element name="audioLanguage" type="xsd:string"/>
                <xsd:element name="subtitlesLanguage" type=" ↩
                   xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="resetSubscriptionRequest">
        <xsd:annotation>
            <xsd:documentation>
                Resets all settings of given subscription.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="addressee">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element maxOccurs="unbounded" name=" ↩
                               subscriptionCode" type=" ↩
                               provisioning:Code"/>
                        </xsd:sequence>
```

```
                </xsd:complexType>
            </xsd:element>
            <xsd:element minOccurs="1" name="defaultMasterPin"
               type="provisioning:Pin"/>
            <xsd:element minOccurs="1" name="defaultParentalPin"
               type="provisioning:Pin"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="sendStbMessageRequest">
    <xsd:annotation>
        <xsd:documentation>Puts message into all subscription's
           STB accounts message queue.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="addressee">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" name="
                           subscriptionCode" type="
                           provisioning:Code"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="messageType" type="
               provisioning:MessageType"/>
            <xsd:element minOccurs="0" name="messageCode" type="
               provisioning:Code"/>
            <xsd:element name="startTime" type="common:DateTime"/
               >
            <xsd:element name="expireSeconds" type="xsd:int"/>
            <xsd:element name="stbOn" type="xsd:boolean">
                <xsd:annotation>
                    <xsd:documentation>Sends the message on every
                        STB startup until it is not expired.</
                       xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name
               ="param" type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>List of parameters.
                       Example: &lt;param&gt;paramName&lt;/param&
                       gt; &lt;param&gt;paramValue&lt;/param&gt;<
                       /xsd:documentation>
                </xsd:annotation>
            </xsd:element>
```

```
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="cancelMessageRequest">
        <xsd:annotation>
            <xsd:documentation>Cancels message from subscription STB ↩
                account's message queue.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="messageCode" type=" ↩
                    provisioning:Code"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="cancelStbOnMessagesRequest">
        <xsd:annotation>
            <xsd:documentation>Cancels all messages for given ↩
                subscription that have stbOn flag. (cancels messages ↩
                for all subscription's stb accounts)</ ↩
                xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionCode" type=" ↩
                    provisioning:Code"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="sendStbAccountMessageRequest">
        <xsd:annotation>
            <xsd:documentation>Puts message into subscription STB ↩
                account's message queue.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="addressee">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element maxOccurs="unbounded" name=" ↩
                                subscriptionStbAccountCode" type=" ↩
                                provisioning:Code"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
```

```
            <xsd:element name="messageType" type=" ↩
               provisioning:MessageType"/>
            <xsd:element minOccurs="0" name="messageCode" type=" ↩
               provisioning:Code"/>
            <xsd:element name="startTime" type="common:DateTime ↩
               "/>
            <xsd:element name="expireSeconds" type="xsd:int"/>
            <xsd:element name="stbOn" type="xsd:boolean">
                <xsd:annotation>
                    <xsd:documentation>Sends the message on every ↩
                        STB startup until it is not expired.</ ↩
                       xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name ↩
               ="param" type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>List of parameters.  ↩
                       Example: &lt;param&gt;paramName&lt;/param& ↩
                       gt; &lt;param&gt;paramValue&lt;/param&gt ↩
                       ;</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="cancelStbAccountStbOnMessagesRequest">
    <xsd:annotation>
        <xsd:documentation>Cancels all messages for given  ↩
           subscription STB account that have stbOn flag.</ ↩
           xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionStbAccountCode" type=" ↩
               provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="rebootStbRequest">
    <xsd:annotation>
        <xsd:documentation>Reboots STB of given subscription STB  ↩
           accounts.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element maxOccurs="unbounded" name=" ↩
                subscriptionStbAccountCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="rebootAllStbRequest">
    <xsd:annotation>
        <xsd:documentation>Reboots all active STB where  ↩
            subscription is billingable.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="portalReloadRequest">
    <xsd:annotation>
        <xsd:documentation>Reloads portal of given subscription  ↩
            STB accounts.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" name=" ↩
                subscriptionStbAccountCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="portalReloadAllRequest">
    <xsd:annotation>
        <xsd:documentation>Reloads portal of all active  ↩
            subscription STB accounts where subscription is  ↩
            billingable.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="createStbAccountRequest">
    <xsd:annotation>
        <xsd:documentation>Creates new subscription STB account  ↩
            for given subscription.</xsd:documentation>
    </xsd:annotation>
```

```
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="common:SchedulableRequest">
                    <xsd:sequence>
                        <xsd:element name="subscriptionCode" type=" ↩
                            provisioning:Code"/>
                        <xsd:element name="subscriptionStbAccountCode ↩
                            " type="provisioning:Code"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="enableStbAccountRequest">
        <xsd:annotation>
            <xsd:documentation>Enables activation of an existing  ↩
               subscription STB account.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="common:SchedulableRequest">
                    <xsd:sequence>
                        <xsd:element name="subscriptionStbAccountCode ↩
                            " type="provisioning:Code"/>
                        <xsd:element minOccurs="0" name="puk" type=" ↩
                            provisioning:Puk"/>
                        <xsd:element minOccurs="0" name="allowedIps"  ↩
                            type="xsd:string">
                             <xsd:annotation>
                                 <xsd:documentation>List of allowed  ↩
                                    IPs when SystemConfig.anyIpAllowed ↩
                                     is disabled. Multiple IP  ↩
                                    addresses could be separated by  ↩
                                    space, comma or semicolon.</ ↩
                                    xsd:documentation>
                             </xsd:annotation>
                        </xsd:element>
                        <xsd:element minOccurs="0" name="stb" type=" ↩
                            provisioning:Stb"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="enableStbAccountWithGeneratedPukRequest">
        <xsd:annotation>
```

```
            <xsd:documentation>Generates PUK and enables activation
               of an existing subscription STB account. Generated PUK
                is returned in response.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="tns:enableStbAccountRequest"/>
                <xsd:element name="pukLength">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:int">
                            <xsd:minInclusive value="1"/>
                            <xsd:maxInclusive value="255"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="enableStbAccountWithGeneratedPukResponse">
        <xsd:annotation>
            <xsd:documentation>Holds the used PUK.</xsd:documentation
               >
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="puk" type="provisioning:Puk"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="createEnabledStbAccountRequest">
        <xsd:annotation>
            <xsd:documentation>Creates new enabled subscription STB
               account for given subscription. Combines create(),
               setPuk(), enable()</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="tns:createStbAccountRequest"/>
                <xsd:element minOccurs="0" name="puk" type="
                   provisioning:Puk"/>
                <xsd:element default="8" minOccurs="0" name="
                   pukLength">
                    <xsd:annotation>
                        <xsd:documentation>When no PUK is given, a
                           PUK of this length is generated.</
                           xsd:documentation>
                    </xsd:annotation>
```

```
                <xsd:simpleType>
                    <xsd:restriction base="xsd:int">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="255"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element minOccurs="0" name="allowedIps" type=" ↩
                xsd:string">
                <xsd:annotation>
                    <xsd:documentation>List of allowed IPs when  ↩
                        SystemConfig.anyIpAllowed is disabled.  ↩
                        Multiple IP addresses could be separated  ↩
                        by space, comma or semicolon.</ ↩
                        xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element minOccurs="0" name="stb" type=" ↩
                provisioning:Stb"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="createEnabledStbAccountResponse">
    <xsd:annotation>
        <xsd:documentation>Holds the used PUK.</xsd:documentation ↩
            >
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="puk" type="provisioning:Puk"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="setStbAccountInfoRequest">
    <xsd:annotation>
        <xsd:documentation>Updates PUK and allowed IPs of given  ↩
            subscription STB account.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionStbAccountCode" type=" ↩
                provisioning:Code"/>
            <xsd:element minOccurs="0" name="puk" nillable="true" ↩
                type="provisioning:Puk">
                <xsd:annotation>
                    <xsd:documentation>Personal Unblocking Key  ↩
                        that the customer has to use on  ↩
```

```
                        registration and other actions.</ ↩
                        xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element minOccurs="0" name="allowedIps" type=" ↩
                    xsd:string">
                    <xsd:annotation>
                        <xsd:documentation>List of allowed IPs when  ↩
                            SystemConfig.anyIpAllowed is disabled.  ↩
                            Multiple IP addresses could be separated  ↩
                            by space, comma or semicolon.</ ↩
                            xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="getStbAccountInfoRequest">
        <xsd:annotation>
            <xsd:documentation>Returns information about subscription ↩
                STB account.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionStbAccountCode" type=" ↩
                    provisioning:Code"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="getStbAccountInfoResponse">
        <xsd:annotation>
            <xsd:documentation>Information about requested  ↩
               subscription STB account.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionStbAccountCode" type=" ↩
                    provisioning:Code"/>
                <xsd:element name="subscriptionCode" type=" ↩
                    provisioning:Code"/>
                <xsd:element name="subscriberCode" type=" ↩
                    provisioning:Code"/>
                <xsd:element name="subscriptionStbAccountState" type ↩
                    ="provisioning:SubscriptionStbAccountState"/>
                <xsd:element minOccurs="0" name="puk" type=" ↩
                    provisioning:Puk"/>
```

```
            <xsd:element minOccurs="0" name="allowedIps" type=" ↩
                xsd:string"/>
            <xsd:element minOccurs="0" name="ipAddress" type=" ↩
                xsd:string"/>
            <xsd:element name="createDate" type="common:DateTime ↩
                "/>
            <xsd:element minOccurs="0" name="stb" type=" ↩
                provisioning:Stb"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="removeStbAccountRequest">
    <xsd:annotation>
        <xsd:documentation>Removes subscription stb account.</ ↩
            xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="common:SchedulableRequest">
                <xsd:sequence>
                    <xsd:element name="subscriptionStbAccountCode ↩
                        " type="provisioning:Code"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getSubscriptionStbAccountsRequest">
    <xsd:annotation>
        <xsd:documentation>Returns list of subscription's stb  ↩
            accounts.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="subscriptionCode" type=" ↩
                provisioning:Code"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getSubscriptionStbAccountsResponse">
    <xsd:annotation>
        <xsd:documentation>Lists subscription's stb accounts and  ↩
            their states.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
```

```
                <xsd:element maxOccurs="unbounded" minOccurs="0" name ↩
                   ="subscriptionStbAccounts">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name=" ↩
                               subscriptionStbAccountCode" type=" ↩
                               provisioning:Code"/>
                            <xsd:element name=" ↩
                               subscriptionStbAccountState" type=" ↩
                               provisioning:SubscriptionStbAccountState ↩
                               "/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="setPortalVersionRequest">
        <xsd:annotation>
            <xsd:documentation>Sets subscription's portal version.</ ↩
               xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="addressee">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element maxOccurs="unbounded" name=" ↩
                               subscriptionCode" type=" ↩
                               provisioning:Code"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            <xsd:element name="portalVersion" type=" ↩
               provisioning:PortalVersion"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ↩
   attributeFormDefault="qualified" elementFormDefault="qualified" ↩
   targetNamespace="http://provisioning.jws.admin.web.tve.itonis.net" ↩
   >
<xsd:complexType name="NotFoundException"/>
<xsd:complexType name="ConflictException"/>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ↩
   attributeFormDefault="qualified" elementFormDefault="qualified" ↩
```

```
    targetNamespace="http://itonis.net/tve/schema/admin/provisioning/ ↩
    subscription/ns">
<xsd:element name="NotFoundException" type="ns2:NotFoundException"/>
<xsd:element name="ConflictException" type="ns2:ConflictException"/>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="getSubscriptionStbAccountsResponse">
    <wsdl:part name="getSubscriptionStbAccountsResponse" element=" ↩
       ns1:getSubscriptionStbAccountsResponse"/>
  </wsdl:message>
  <wsdl:message name="cancelMessageResponse">
  </wsdl:message>
  <wsdl:message name="getInfoRequest">
    <wsdl:part name="getInfoRequest" element="ns1:getInfoRequest"/>
  </wsdl:message>
  <wsdl:message name="setSubscriptionStbAccountInfoRequest">
    <wsdl:part name="setStbAccountInfoRequest" element=" ↩
       ns1:setStbAccountInfoRequest"/>
  </wsdl:message>
  <wsdl:message name="setChargeLimitResponse">
  </wsdl:message>
  <wsdl:message name="unassignStbResponse">
  </wsdl:message>
  <wsdl:message name="renewOfferRequest">
    <wsdl:part name="renewOfferRequest" element=" ↩
       ns1:renewOfferRequest"/>
  </wsdl:message>
  <wsdl:message name="unassignStbRequest">
    <wsdl:part name="unassignStbRequest" element=" ↩
       ns1:unassignStbRequest"/>
  </wsdl:message>
  <wsdl:message name="createEnabledSubscriptionStbAccountRequest">
    <wsdl:part name="createEnabledStbAccountRequest" element=" ↩
       ns1:createEnabledStbAccountRequest"/>
  </wsdl:message>
  <wsdl:message name="setInfoResponse">
  </wsdl:message>
  <wsdl:message name="sendSubscriptionStbAccountMessageResponse">
  </wsdl:message>
  <wsdl:message name="enableWithGeneratedPukRequest">
    <wsdl:part name="enableWithGeneratedPukRequest" element=" ↩
       ns1:enableWithGeneratedPukRequest"/>
  </wsdl:message>
  <wsdl:message name="suspendRequest">
    <wsdl:part name="suspendRequest" element="ns1:suspendRequest"/>
  </wsdl:message>
  <wsdl:message name="unsubscribeOfferRequest">
    <wsdl:part name="unsubscribeOfferRequest" element=" ↩
       ns1:unsubscribeOfferRequest"/>
```

```
</wsdl:message>
<wsdl:message name="disableResponse">
</wsdl:message>
<wsdl:message name="rebootAllStbResponse">
</wsdl:message>
<wsdl:message name="unsubscribeOfferResponse">
</wsdl:message>
<wsdl:message name="createSubscriptionStbAccountRequest">
  <wsdl:part name="createStbAccountRequest" element=" ↩
     ns1:createStbAccountRequest"/>
</wsdl:message>
<wsdl:message name="sendSubscriptionStbAccountMessageRequest">
  <wsdl:part name="sendStbAccountMessageRequest" element=" ↩
     ns1:sendStbAccountMessageRequest"/>
</wsdl:message>
<wsdl:message name="disableTvodRequest">
  <wsdl:part name="disableTvodRequest" element=" ↩
     ns1:disableTvodRequest"/>
</wsdl:message>
<wsdl:message name="setPreferredAudioSubtitlesResponse">
</wsdl:message>
<wsdl:message name="portalReloadRequest">
  <wsdl:part name="portalReloadRequest" element=" ↩
     ns1:portalReloadRequest"/>
</wsdl:message>
<wsdl:message name=" ↩
   enableSubscriptionStbAccountWithGeneratedPukResponse">
  <wsdl:part name="enableStbAccountWithGeneratedPukResponse"  ↩
     element="ns1:enableStbAccountWithGeneratedPukResponse"/>
</wsdl:message>
<wsdl:message name="changeTariffRequest">
  <wsdl:part name="changeTariffRequest" element=" ↩
     ns1:changeTariffRequest"/>
</wsdl:message>
<wsdl:message name="ConflictException">
  <wsdl:part name="ConflictException" element=" ↩
     tns:ConflictException"/>
</wsdl:message>
<wsdl:message name="portalReloadResponse">
</wsdl:message>
<wsdl:message name="setChargeLimitRequest">
  <wsdl:part name="setChargeLimitRequest" element=" ↩
     ns1:setChargeLimitRequest"/>
</wsdl:message>
<wsdl:message name="setLocalityResponse">
</wsdl:message>
<wsdl:message name="disableRequest">
  <wsdl:part name="disableRequest" element="ns1:disableRequest"/>
</wsdl:message>
```

```
<wsdl:message name="setPreferredAudioSubtitlesRequest">
  <wsdl:part name="setPreferredAudioSubtitlesRequest" element=" ↩
     ns1:setPreferredAudioSubtitlesRequest"/>
</wsdl:message>
<wsdl:message name="createEnabledSubscriptionStbAccountResponse">
  <wsdl:part name="createEnabledStbAccountResponse" element=" ↩
     ns1:createEnabledStbAccountResponse"/>
</wsdl:message>
<wsdl:message name="getChargeLimitResponse">
  <wsdl:part name="getChargeLimitResponse" element=" ↩
     ns1:getChargeLimitResponse"/>
</wsdl:message>
<wsdl:message name="setParentalPinResponse">
</wsdl:message>
<wsdl:message name=" ↩
   cancelSubscriptionStbAccountStbOnMessagesResponse">
</wsdl:message>
<wsdl:message name="enableRequest">
  <wsdl:part name="enableRequest" element="ns1:enableRequest"/>
</wsdl:message>
<wsdl:message name="setMasterPinResponse">
</wsdl:message>
<wsdl:message name="getInfoResponse">
  <wsdl:part name="getInfoResponse" element="ns1:getInfoResponse"/>
</wsdl:message>
<wsdl:message name="getOffersResponse">
  <wsdl:part name="getOffersResponse" element=" ↩
     ns1:getOffersResponse"/>
</wsdl:message>
<wsdl:message name="NotFoundException">
  <wsdl:part name="NotFoundException" element=" ↩
     tns:NotFoundException"/>
</wsdl:message>
<wsdl:message name="rebootAllStbRequest">
  <wsdl:part name="rebootAllStbRequest" element=" ↩
     ns1:rebootAllStbRequest"/>
</wsdl:message>
<wsdl:message name="assignStbRequest">
  <wsdl:part name="assignStbRequest" element="ns1:assignStbRequest" ↩
     />
</wsdl:message>
<wsdl:message name="sendStbMessageResponse">
</wsdl:message>
<wsdl:message name="subscribeOfferResponse">
</wsdl:message>
<wsdl:message name="cancelStbOnMessagesRequest">
  <wsdl:part name="cancelStbOnMessagesRequest" element=" ↩
     ns1:cancelStbOnMessagesRequest"/>
</wsdl:message>
```

```
<wsdl:message name="enableSubscriptionStbAccountResponse">
</wsdl:message>
<wsdl:message name="cancelStbOnMessagesResponse">
</wsdl:message>
<wsdl:message name=" ↩
   enableSubscriptionStbAccountWithGeneratedPukRequest">
  <wsdl:part name="enableStbAccountWithGeneratedPukRequest" element ↩
     ="ns1:enableStbAccountWithGeneratedPukRequest"/>
</wsdl:message>
<wsdl:message name="suspendResponse">
</wsdl:message>
<wsdl:message name="assignStbResponse">
</wsdl:message>
<wsdl:message name="setSubscriptionStbAccountInfoResponse">
</wsdl:message>
<wsdl:message name="createEnabledResponse">
  <wsdl:part name="createEnabledResponse" element=" ↩
     ns1:createEnabledResponse"/>
</wsdl:message>
<wsdl:message name="changeTariffResponse">
</wsdl:message>
<wsdl:message name="enableTvodResponse">
</wsdl:message>
<wsdl:message name="setLocalityRequest">
  <wsdl:part name="setLocalityRequest" element=" ↩
     ns1:setLocalityRequest"/>
</wsdl:message>
<wsdl:message name=" ↩
   cancelSubscriptionStbAccountStbOnMessagesRequest">
  <wsdl:part name="cancelStbAccountStbOnMessagesRequest" element=" ↩
     ns1:cancelStbAccountStbOnMessagesRequest"/>
</wsdl:message>
<wsdl:message name="getStbInfoResponse">
  <wsdl:part name="getStbInfoResponse" element=" ↩
     ns1:getStbInfoResponse"/>
</wsdl:message>
<wsdl:message name="renewOfferResponse">
</wsdl:message>
<wsdl:message name="portalReloadAllResponse">
</wsdl:message>
<wsdl:message name="rebootStbResponse">
</wsdl:message>
<wsdl:message name="createSubscriptionStbAccountResponse">
</wsdl:message>
<wsdl:message name="createRequest">
  <wsdl:part name="createRequest" element="ns1:createRequest"/>
</wsdl:message>
<wsdl:message name="enableWithGeneratedPukResponse">
```

```
  <wsdl:part name="enableWithGeneratedPukResponse" element=" ↩
     ns1:enableWithGeneratedPukResponse"/>
</wsdl:message>
<wsdl:message name="sendStbMessageRequest">
  <wsdl:part name="sendStbMessageRequest" element=" ↩
     ns1:sendStbMessageRequest"/>
</wsdl:message>
<wsdl:message name="enableTvodRequest">
  <wsdl:part name="enableTvodRequest" element=" ↩
     ns1:enableTvodRequest"/>
</wsdl:message>
<wsdl:message name="setPortalVersionRequest">
  <wsdl:part name="setPortalVersionRequest" element=" ↩
     ns1:setPortalVersionRequest"/>
</wsdl:message>
<wsdl:message name="createWithoutStbAccountResponse">
</wsdl:message>
<wsdl:message name="setMasterPinRequest">
  <wsdl:part name="setMasterPinRequest" element=" ↩
     ns1:setMasterPinRequest"/>
</wsdl:message>
<wsdl:message name="enableSubscriptionStbAccountRequest">
  <wsdl:part name="enableStbAccountRequest" element=" ↩
     ns1:enableStbAccountRequest"/>
</wsdl:message>
<wsdl:message name="getStbInfoRequest">
  <wsdl:part name="getStbInfoRequest" element=" ↩
     ns1:getStbInfoRequest"/>
</wsdl:message>
<wsdl:message name="setInfoRequest">
  <wsdl:part name="setInfoRequest" element="ns1:setInfoRequest"/>
</wsdl:message>
<wsdl:message name="setParentalPinRequest">
  <wsdl:part name="setParentalPinRequest" element=" ↩
     ns1:setParentalPinRequest"/>
</wsdl:message>
<wsdl:message name="getSubscriptionStbAccountInfoResponse">
  <wsdl:part name="getStbAccountInfoResponse" element=" ↩
     ns1:getStbAccountInfoResponse"/>
</wsdl:message>
<wsdl:message name="removeSubscriptionStbAccountRequest">
  <wsdl:part name="removeStbAccountRequest" element=" ↩
     ns1:removeStbAccountRequest"/>
</wsdl:message>
<wsdl:message name="getSubscriptionStbAccountInfoRequest">
  <wsdl:part name="getStbAccountInfoRequest" element=" ↩
     ns1:getStbAccountInfoRequest"/>
</wsdl:message>
<wsdl:message name="createResponse">
```

```
</wsdl:message>
<wsdl:message name="getOffersRequest">
  <wsdl:part name="getOffersRequest" element="ns1:getOffersRequest" ↩
     />
</wsdl:message>
<wsdl:message name="subscribeOfferRequest">
  <wsdl:part name="subscribeOfferRequest" element=" ↩
     ns1:subscribeOfferRequest"/>
</wsdl:message>
<wsdl:message name="getSubscriptionStbAccountsRequest">
  <wsdl:part name="getSubscriptionStbAccountsRequest" element=" ↩
     ns1:getSubscriptionStbAccountsRequest"/>
</wsdl:message>
<wsdl:message name="createEnabledRequest">
  <wsdl:part name="createEnabledRequest" element=" ↩
     ns1:createEnabledRequest"/>
</wsdl:message>
<wsdl:message name="disableTvodResponse">
</wsdl:message>
<wsdl:message name="portalReloadAllRequest">
  <wsdl:part name="portalReloadAllRequest" element=" ↩
     ns1:portalReloadAllRequest"/>
</wsdl:message>
<wsdl:message name="createWithoutStbAccountRequest">
  <wsdl:part name="createWithoutStbAccountRequest" element=" ↩
     ns1:createWithoutStbAccountRequest"/>
</wsdl:message>
<wsdl:message name="setPortalVersionResponse">
</wsdl:message>
<wsdl:message name="cancelMessageRequest">
  <wsdl:part name="cancelMessageRequest" element=" ↩
     ns1:cancelMessageRequest"/>
</wsdl:message>
<wsdl:message name="enableResponse">
</wsdl:message>
<wsdl:message name="resetSubscriptionResponse">
</wsdl:message>
<wsdl:message name="removeSubscriptionStbAccountResponse">
</wsdl:message>
<wsdl:message name="getChargeLimitRequest">
  <wsdl:part name="getChargeLimitRequest" element=" ↩
     ns1:getChargeLimitRequest"/>
</wsdl:message>
<wsdl:message name="rebootStbRequest">
  <wsdl:part name="rebootStbRequest" element="ns1:rebootStbRequest" ↩
     />
</wsdl:message>
<wsdl:message name="resetSubscriptionRequest">
```

```
    <wsdl:part name="resetSubscriptionRequest" element=" ↩
       ns1:resetSubscriptionRequest"/>
  </wsdl:message>
  <wsdl:portType name="SubscriptionEndpointPortType">
    <wsdl:operation name="removeSubscriptionStbAccount">
      <wsdl:input name="removeSubscriptionStbAccountRequest" message= ↩
         "tns:removeSubscriptionStbAccountRequest"/>
      <wsdl:output name="removeSubscriptionStbAccountResponse"  ↩
         message="tns:removeSubscriptionStbAccountResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="portalReload">
      <wsdl:input name="portalReloadRequest" message=" ↩
         tns:portalReloadRequest"/>
      <wsdl:output name="portalReloadResponse" message=" ↩
         tns:portalReloadResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="getStbInfo">
      <wsdl:input name="getStbInfoRequest" message=" ↩
         tns:getStbInfoRequest"/>
      <wsdl:output name="getStbInfoResponse" message=" ↩
         tns:getStbInfoResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="rebootStb">
      <wsdl:input name="rebootStbRequest" message=" ↩
         tns:rebootStbRequest"/>
      <wsdl:output name="rebootStbResponse" message=" ↩
         tns:rebootStbResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="rebootAllStb">
      <wsdl:input name="rebootAllStbRequest" message=" ↩
         tns:rebootAllStbRequest"/>
      <wsdl:output name="rebootAllStbResponse" message=" ↩
         tns:rebootAllStbResponse"/>
    </wsdl:operation>
    <wsdl:operation name="sendStbMessage">
      <wsdl:input name="sendStbMessageRequest" message=" ↩
         tns:sendStbMessageRequest"/>
```

```
      <wsdl:output name="sendStbMessageResponse" message=" ↩
         tns:sendStbMessageResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="createWithoutStbAccount">
      <wsdl:input name="createWithoutStbAccountRequest" message=" ↩
         tns:createWithoutStbAccountRequest"/>
      <wsdl:output name="createWithoutStbAccountResponse" message=" ↩
         tns:createWithoutStbAccountResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="changeTariff">
      <wsdl:input name="changeTariffRequest" message=" ↩
         tns:changeTariffRequest"/>
      <wsdl:output name="changeTariffResponse" message=" ↩
         tns:changeTariffResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="cancelSubscriptionStbAccountStbOnMessages">
      <wsdl:input name=" ↩
         cancelSubscriptionStbAccountStbOnMessagesRequest" message=" ↩
         tns:cancelSubscriptionStbAccountStbOnMessagesRequest"/>
      <wsdl:output name=" ↩
         cancelSubscriptionStbAccountStbOnMessagesResponse" message=" ↩
         tns:cancelSubscriptionStbAccountStbOnMessagesResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="resetSubscription">
      <wsdl:input name="resetSubscriptionRequest" message=" ↩
         tns:resetSubscriptionRequest"/>
      <wsdl:output name="resetSubscriptionResponse" message=" ↩
         tns:resetSubscriptionResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="portalReloadAll">
      <wsdl:input name="portalReloadAllRequest" message=" ↩
         tns:portalReloadAllRequest"/>
```

```
    <wsdl:output name="portalReloadAllResponse" message=" ↩
       tns:portalReloadAllResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getChargeLimit">
    <wsdl:input name="getChargeLimitRequest" message=" ↩
       tns:getChargeLimitRequest"/>
    <wsdl:output name="getChargeLimitResponse" message=" ↩
       tns:getChargeLimitResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="setParentalPin">
    <wsdl:input name="setParentalPinRequest" message=" ↩
       tns:setParentalPinRequest"/>
    <wsdl:output name="setParentalPinResponse" message=" ↩
       tns:setParentalPinResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
    <wsdl:fault name="ConflictException" message=" ↩
       tns:ConflictException"/>
  </wsdl:operation>
  <wsdl:operation name="renewOffer">
    <wsdl:input name="renewOfferRequest" message=" ↩
       tns:renewOfferRequest"/>
    <wsdl:output name="renewOfferResponse" message=" ↩
       tns:renewOfferResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
    <wsdl:fault name="ConflictException" message=" ↩
       tns:ConflictException"/>
  </wsdl:operation>
  <wsdl:operation name="enableTvod">
    <wsdl:input name="enableTvodRequest" message=" ↩
       tns:enableTvodRequest"/>
    <wsdl:output name="enableTvodResponse" message=" ↩
       tns:enableTvodResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="getSubscriptionStbAccounts">
    <wsdl:input name="getSubscriptionStbAccountsRequest" message=" ↩
       tns:getSubscriptionStbAccountsRequest"/>
    <wsdl:output name="getSubscriptionStbAccountsResponse" message= ↩
       "tns:getSubscriptionStbAccountsResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="enableWithGeneratedPuk">
```

```
      <wsdl:input name="enableWithGeneratedPukRequest" message=" ↩
         tns:enableWithGeneratedPukRequest"/>
      <wsdl:output name="enableWithGeneratedPukResponse" message=" ↩
         tns:enableWithGeneratedPukResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="createEnabled">
      <wsdl:input name="createEnabledRequest" message=" ↩
         tns:createEnabledRequest"/>
      <wsdl:output name="createEnabledResponse" message=" ↩
         tns:createEnabledResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="create">
      <wsdl:input name="createRequest" message="tns:createRequest"/>
      <wsdl:output name="createResponse" message="tns:createResponse" ↩
         />
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="getInfo">
      <wsdl:input name="getInfoRequest" message="tns:getInfoRequest"/ ↩
         >
      <wsdl:output name="getInfoResponse" message=" ↩
         tns:getInfoResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="setInfo">
      <wsdl:input name="setInfoRequest" message="tns:setInfoRequest"/ ↩
         >
      <wsdl:output name="setInfoResponse" message=" ↩
         tns:setInfoResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="setLocality">
      <wsdl:input name="setLocalityRequest" message=" ↩
         tns:setLocalityRequest"/>
```

```
    <wsdl:output name="setLocalityResponse" message=" ↩
       tns:setLocalityResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="cancelMessage">
    <wsdl:input name="cancelMessageRequest" message=" ↩
       tns:cancelMessageRequest"/>
    <wsdl:output name="cancelMessageResponse" message=" ↩
       tns:cancelMessageResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="setSubscriptionStbAccountInfo">
    <wsdl:input name="setSubscriptionStbAccountInfoRequest" message ↩
       ="tns:setSubscriptionStbAccountInfoRequest"/>
    <wsdl:output name="setSubscriptionStbAccountInfoResponse"  ↩
       message="tns:setSubscriptionStbAccountInfoResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
    <wsdl:fault name="ConflictException" message=" ↩
       tns:ConflictException"/>
  </wsdl:operation>
  <wsdl:operation name="createEnabledSubscriptionStbAccount">
    <wsdl:input name="createEnabledSubscriptionStbAccountRequest"  ↩
       message="tns:createEnabledSubscriptionStbAccountRequest"/>
    <wsdl:output name="createEnabledSubscriptionStbAccountResponse" ↩
        message="tns:createEnabledSubscriptionStbAccountResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
    <wsdl:fault name="ConflictException" message=" ↩
       tns:ConflictException"/>
  </wsdl:operation>
  <wsdl:operation name="setPortalVersion">
    <wsdl:input name="setPortalVersionRequest" message=" ↩
       tns:setPortalVersionRequest"/>
    <wsdl:output name="setPortalVersionResponse" message=" ↩
       tns:setPortalVersionResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="setPreferredAudioSubtitles">
    <wsdl:input name="setPreferredAudioSubtitlesRequest" message=" ↩
       tns:setPreferredAudioSubtitlesRequest"/>
    <wsdl:output name="setPreferredAudioSubtitlesResponse" message= ↩
       "tns:setPreferredAudioSubtitlesResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
```

```
    <wsdl:operation name="unassignStb">
      <wsdl:input name="unassignStbRequest" message=" ↩
         tns:unassignStbRequest"/>
      <wsdl:output name="unassignStbResponse" message=" ↩
         tns:unassignStbResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="disableTvod">
      <wsdl:input name="disableTvodRequest" message=" ↩
         tns:disableTvodRequest"/>
      <wsdl:output name="disableTvodResponse" message=" ↩
         tns:disableTvodResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="enableSubscriptionStbAccount">
      <wsdl:input name="enableSubscriptionStbAccountRequest" message= ↩
         "tns:enableSubscriptionStbAccountRequest"/>
      <wsdl:output name="enableSubscriptionStbAccountResponse"  ↩
         message="tns:enableSubscriptionStbAccountResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="createSubscriptionStbAccount">
      <wsdl:input name="createSubscriptionStbAccountRequest" message= ↩
         "tns:createSubscriptionStbAccountRequest"/>
      <wsdl:output name="createSubscriptionStbAccountResponse"  ↩
         message="tns:createSubscriptionStbAccountResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="assignStb">
      <wsdl:input name="assignStbRequest" message=" ↩
         tns:assignStbRequest"/>
      <wsdl:output name="assignStbResponse" message=" ↩
         tns:assignStbResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="getSubscriptionStbAccountInfo">
```

```
    <wsdl:input name="getSubscriptionStbAccountInfoRequest" message ↩
       ="tns:getSubscriptionStbAccountInfoRequest"/>
    <wsdl:output name="getSubscriptionStbAccountInfoResponse"  ↩
       message="tns:getSubscriptionStbAccountInfoResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="subscribeOffer">
    <wsdl:input name="subscribeOfferRequest" message=" ↩
       tns:subscribeOfferRequest"/>
    <wsdl:output name="subscribeOfferResponse" message=" ↩
       tns:subscribeOfferResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
    <wsdl:fault name="ConflictException" message=" ↩
       tns:ConflictException"/>
  </wsdl:operation>
  <wsdl:operation name="setChargeLimit">
    <wsdl:input name="setChargeLimitRequest" message=" ↩
       tns:setChargeLimitRequest"/>
    <wsdl:output name="setChargeLimitResponse" message=" ↩
       tns:setChargeLimitResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="disable">
    <wsdl:input name="disableRequest" message="tns:disableRequest"/ ↩
       >
    <wsdl:output name="disableResponse" message=" ↩
       tns:disableResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name=" ↩
     enableSubscriptionStbAccountWithGeneratedPuk">
    <wsdl:input name=" ↩
       enableSubscriptionStbAccountWithGeneratedPukRequest" message ↩
       ="tns:enableSubscriptionStbAccountWithGeneratedPukRequest"/>
    <wsdl:output name=" ↩
       enableSubscriptionStbAccountWithGeneratedPukResponse"  ↩
       message=" ↩
       tns:enableSubscriptionStbAccountWithGeneratedPukResponse"/>
    <wsdl:fault name="NotFoundException" message=" ↩
       tns:NotFoundException"/>
    <wsdl:fault name="ConflictException" message=" ↩
       tns:ConflictException"/>
  </wsdl:operation>
  <wsdl:operation name="setMasterPin">
```

```
      <wsdl:input name="setMasterPinRequest" message=" ↩
         tns:setMasterPinRequest"/>
      <wsdl:output name="setMasterPinResponse" message=" ↩
         tns:setMasterPinResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="unsubscribeOffer">
      <wsdl:input name="unsubscribeOfferRequest" message=" ↩
         tns:unsubscribeOfferRequest"/>
      <wsdl:output name="unsubscribeOfferResponse" message=" ↩
         tns:unsubscribeOfferResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="getOffers">
      <wsdl:input name="getOffersRequest" message=" ↩
         tns:getOffersRequest"/>
      <wsdl:output name="getOffersResponse" message=" ↩
         tns:getOffersResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="suspend">
      <wsdl:input name="suspendRequest" message="tns:suspendRequest"/ ↩
         >
      <wsdl:output name="suspendResponse" message=" ↩
         tns:suspendResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
    <wsdl:operation name="cancelStbOnMessages">
      <wsdl:input name="cancelStbOnMessagesRequest" message=" ↩
         tns:cancelStbOnMessagesRequest"/>
      <wsdl:output name="cancelStbOnMessagesResponse" message=" ↩
         tns:cancelStbOnMessagesResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="sendSubscriptionStbAccountMessage">
      <wsdl:input name="sendSubscriptionStbAccountMessageRequest"  ↩
         message="tns:sendSubscriptionStbAccountMessageRequest"/>
```

```
      <wsdl:output name="sendSubscriptionStbAccountMessageResponse"  ↩
         message="tns:sendSubscriptionStbAccountMessageResponse"/>
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
    </wsdl:operation>
    <wsdl:operation name="enable">
      <wsdl:input name="enableRequest" message="tns:enableRequest"/>
      <wsdl:output name="enableResponse" message="tns:enableResponse" ↩
         />
      <wsdl:fault name="NotFoundException" message=" ↩
         tns:NotFoundException"/>
      <wsdl:fault name="ConflictException" message=" ↩
         tns:ConflictException"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SubscriptionEndpointHttpBinding" type=" ↩
     tns:SubscriptionEndpointPortType">
    <wsdlsoap:binding style="document" transport="http://schemas. ↩
       xmlsoap.org/soap/http"/>
    <wsdl:operation name="removeSubscriptionStbAccount">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="removeSubscriptionStbAccountRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="removeSubscriptionStbAccountResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="portalReload">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="portalReloadRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="portalReloadResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getStbInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getStbInfoRequest">
```

```
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getStbInfoResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConflictException">
      <wsdlsoap:fault name="ConflictException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="rebootStb">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="rebootStbRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="rebootStbResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="rebootAllStb">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="rebootAllStbRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="rebootAllStbResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="sendStbMessage">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="sendStbMessageRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="sendStbMessageResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="createWithoutStbAccount">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="createWithoutStbAccountRequest">
      <wsdlsoap:body use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output name="createWithoutStbAccountResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="changeTariff">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="changeTariffRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="changeTariffResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="cancelSubscriptionStbAccountStbOnMessages">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name=" ↩
         cancelSubscriptionStbAccountStbOnMessagesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name=" ↩
         cancelSubscriptionStbAccountStbOnMessagesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="resetSubscription">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="resetSubscriptionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="resetSubscriptionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
```

```
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConflictException">
      <wsdlsoap:fault name="ConflictException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="portalReloadAll">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="portalReloadAllRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="portalReloadAllResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getChargeLimit">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="getChargeLimitRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getChargeLimitResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="setParentalPin">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="setParentalPinRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="setParentalPinResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConflictException">
      <wsdlsoap:fault name="ConflictException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="renewOffer">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="renewOfferRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="renewOfferResponse">
      <wsdlsoap:body use="literal"/>
```

```
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="enableTvod">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="enableTvodRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="enableTvodResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getSubscriptionStbAccounts">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSubscriptionStbAccountsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSubscriptionStbAccountsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="enableWithGeneratedPuk">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="enableWithGeneratedPukRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="enableWithGeneratedPukResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createEnabled">
      <wsdlsoap:operation soapAction=""/>
```

```
      <wsdl:input name="createEnabledRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createEnabledResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="create">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="createRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getInfoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getInfoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setInfoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setInfoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
```

```
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConflictException">
      <wsdlsoap:fault name="ConflictException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="setLocality">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="setLocalityRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="setLocalityResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="cancelMessage">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="cancelMessageRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="cancelMessageResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="setSubscriptionStbAccountInfo">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="setSubscriptionStbAccountInfoRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="setSubscriptionStbAccountInfoResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundException">
      <wsdlsoap:fault name="NotFoundException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConflictException">
      <wsdlsoap:fault name="ConflictException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="createEnabledSubscriptionStbAccount">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="createEnabledSubscriptionStbAccountRequest">
      <wsdlsoap:body use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output name="createEnabledSubscriptionStbAccountResponse" ↩
         >
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setPortalVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setPortalVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setPortalVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setPreferredAudioSubtitles">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setPreferredAudioSubtitlesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setPreferredAudioSubtitlesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="unassignStb">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="unassignStbRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="unassignStbResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
```

```
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="disableTvod">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="disableTvodRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="disableTvodResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="enableSubscriptionStbAccount">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="enableSubscriptionStbAccountRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="enableSubscriptionStbAccountResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createSubscriptionStbAccount">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="createSubscriptionStbAccountRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createSubscriptionStbAccountResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="assignStb">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="assignStbRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
```

```
      <wsdl:output name="assignStbResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getSubscriptionStbAccountInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSubscriptionStbAccountInfoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSubscriptionStbAccountInfoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="subscribeOffer">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="subscribeOfferRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="subscribeOfferResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setChargeLimit">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setChargeLimitRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setChargeLimitResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
```

```
    <wsdl:operation name="disable">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="disableRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="disableResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name=" ↩
       enableSubscriptionStbAccountWithGeneratedPuk">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name=" ↩
          enableSubscriptionStbAccountWithGeneratedPukRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name=" ↩
          enableSubscriptionStbAccountWithGeneratedPukResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setMasterPin">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="setMasterPinRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setMasterPinResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="unsubscribeOffer">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="unsubscribeOfferRequest">
        <wsdlsoap:body use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output name="unsubscribeOfferResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getOffers">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getOffersRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getOffersResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="suspend">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="suspendRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="suspendResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="cancelStbOnMessages">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="cancelStbOnMessagesRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="cancelStbOnMessagesResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
```

```
    </wsdl:operation>
    <wsdl:operation name="sendSubscriptionStbAccountMessage">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="sendSubscriptionStbAccountMessageRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="sendSubscriptionStbAccountMessageResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="enable">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="enableRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="enableResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotFoundException">
        <wsdlsoap:fault name="NotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConflictException">
        <wsdlsoap:fault name="ConflictException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SubscriptionEndpoint">
    <wsdl:port name="SubscriptionEndpointHttpPort" binding=" ↩
       tns:SubscriptionEndpointHttpBinding">
      <wsdlsoap:address location="http://localhost/services/ ↩
         SubscriptionEndpoint"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

1.6.3 WSDL CRM Offer management

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://itonis.net/tve/schema/crm/ ↩
   subscription/ns" xmlns:tns="http://itonis.net/tve/schema/crm/ ↩
   subscription/ns" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/ ↩
   soap/" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"  ↩
   xmlns:ns1="http://itonis.net/tve/schema/crm/subscription"  ↩
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenc11=" ↩
   http://schemas.xmlsoap.org/soap/encoding/" xmlns:soapenc12="http: ↩
```

```
   //www.w3.org/2003/05/soap-encoding" xmlns:soap11="http://schemas. ↩
   xmlsoap.org/soap/envelope/" xmlns:wsdl="http://schemas.xmlsoap.org ↩
   /wsdl/">
  <wsdl:types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns=" ↩
   http://itonis.net/tve/schema/crm/subscription" elementFormDefault= ↩
   "unqualified" targetNamespace="http://itonis.net/tve/schema/crm/ ↩
   subscription">

    <xsd:simpleType name="Code">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="32"/>
        </xsd:restriction>
    </xsd:simpleType>

     <xsd:complexType name="FaultDetail">
        <xsd:sequence>
            <xsd:element name="message" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="crmSubscribeOfferRequest">
        <xsd:annotation>
            <xsd:documentation>Tells the CRM system to subscribe  ↩
               offer for subscription.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionCode" type="tns:Code"/ ↩
                   >
                <xsd:element name="offerCode" type="tns:Code"/>
                <xsd:element name="ceil" type="xsd:boolean">
                    <xsd:annotation>
                        <xsd:documentation>Whether subscribing is  ↩
                           immediate (false), or since the next  ↩
                           billing period (true).</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="crmSubscribeOfferResponse">
        <xsd:annotation>
            <xsd:documentation>Response from CRM system for  ↩
               subscribeOfferRequest.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
```

```
                <xsd:element name="subscribeDate" type="xsd:date"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="crmUnsubscribeOfferRequest">
        <xsd:annotation>
            <xsd:documentation>Tells the CRM system to unsubscribe ↩
               offer for subscription.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionCode" type="tns:Code"/ ↩
                   >
                <xsd:element name="offerCode" type="tns:Code"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="crmUnsubscribeOfferResponse">
        <xsd:annotation>
            <xsd:documentation>Response from CRM system for ↩
               unsubscribeOfferRequest.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="unsubscribeDate" type="xsd:date"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ↩
   attributeFormDefault="qualified" elementFormDefault="qualified" ↩
   targetNamespace="http://itonis.net/tve/schema/crm/subscription/ns" ↩
   >
<xsd:element name="crmConflictException" type="ns1:FaultDetail"/>
<xsd:element name="crmNotFoundException" type="ns1:FaultDetail"/>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="crmSubscribeOfferRequest">
    <wsdl:part name="crmSubscribeOfferRequest" element=" ↩
       ns1:crmSubscribeOfferRequest"/>
  </wsdl:message>
  <wsdl:message name="crmUnsubscribeOfferRequest">
    <wsdl:part name="crmUnsubscribeOfferRequest" element=" ↩
       ns1:crmUnsubscribeOfferRequest"/>
  </wsdl:message>
  <wsdl:message name="crmConflictException">
    <wsdl:part name="crmConflictException" element=" ↩
       tns:crmConflictException"/>
```

```
</wsdl:message>
<wsdl:message name="crmUnsubscribeOfferResponse">
  <wsdl:part name="crmUnsubscribeOfferResponse" element=" ↩
     ns1:crmUnsubscribeOfferResponse"/>
</wsdl:message>
<wsdl:message name="crmSubscribeOfferResponse">
  <wsdl:part name="crmSubscribeOfferResponse" element=" ↩
     ns1:crmSubscribeOfferResponse"/>
</wsdl:message>
<wsdl:message name="crmNotFoundException">
  <wsdl:part name="crmNotFoundException" element=" ↩
     tns:crmNotFoundException"/>
</wsdl:message>
<wsdl:portType name="CrmOfferEndpointPortType">
  <wsdl:operation name="crmSubscribeOffer">
    <wsdl:input name="crmSubscribeOfferRequest" message=" ↩
       tns:crmSubscribeOfferRequest"/>
    <wsdl:output name="crmSubscribeOfferResponse" message=" ↩
       tns:crmSubscribeOfferResponse"/>
    <wsdl:fault name="crmConflictException" message=" ↩
       tns:crmConflictException"/>
    <wsdl:fault name="crmNotFoundException" message=" ↩
       tns:crmNotFoundException"/>
  </wsdl:operation>
  <wsdl:operation name="crmUnsubscribeOffer">
    <wsdl:input name="crmUnsubscribeOfferRequest" message=" ↩
       tns:crmUnsubscribeOfferRequest"/>
    <wsdl:output name="crmUnsubscribeOfferResponse" message=" ↩
       tns:crmUnsubscribeOfferResponse"/>
    <wsdl:fault name="crmConflictException" message=" ↩
       tns:crmConflictException"/>
    <wsdl:fault name="crmNotFoundException" message=" ↩
       tns:crmNotFoundException"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CrmOfferEndpointHttpBinding" type=" ↩
   tns:CrmOfferEndpointPortType">
  <wsdlsoap:binding style="document" transport="http://schemas. ↩
     xmlsoap.org/soap/http"/>
  <wsdl:operation name="crmSubscribeOffer">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="crmSubscribeOfferRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="crmSubscribeOfferResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="crmConflictException">
      <wsdlsoap:fault name="crmConflictException" use="literal"/>
```

```
      </wsdl:fault>
      <wsdl:fault name="crmNotFoundException">
        <wsdlsoap:fault name="crmNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="crmUnsubscribeOffer">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="crmUnsubscribeOfferRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="crmUnsubscribeOfferResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="crmConflictException">
        <wsdlsoap:fault name="crmConflictException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="crmNotFoundException">
        <wsdlsoap:fault name="crmNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CrmOfferEndpoint">
    <wsdl:port name="CrmOfferEndpointHttpPort" binding=" ↩
       tns:CrmOfferEndpointHttpBinding">
      <wsdlsoap:address location="http://localhost:9090/services/ ↩
         CrmOfferEndpoint"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Chapter 2

Billing Interface

2.1 Introduction

nangu.TV offers modularized billing data export interface. The billing data unit is called CDR (abbrevation for Call Detail Record). There are few modules for CDR output; currently XML and CSV output interface is implemented.

2.2 CDR information

A CDR carries the following information independently of its form:

CDR ID – unique identifier of the CDR.

CDR generation timestamp – determines the time when the CDR was created.

Subscriber ID – platform identifier of the billed subscriber.

Subscriber code – identifier of the subscriber in the provider's CRM.

Subscription ID – platform identifier of the subscription.

Subscription code – identifier of the subscription in the provider's CRM.

Tariff ID – platform identifier of the subscription's tariff.

Tariff code – identifier of the subscription's tariff in the provider's CRM.

CDR type – identification of the billed service:

- **OFFER** – monthly payment for offer,
- **VOD** – payment for single purchased program.

CDR subtype – detail about single purchased program:

- **TVoD** – payment for transactionally purchased program,
- **SVoD** – watching program included in pre-paid package.

Start and end timestamps – timestamps of the beginning and the end of time span the CDR relates to. For offer subscriptions, it is the invoicing period. For VOD transactions, it is the program rental time span.

Units – number of drawn service units. This is only set for TVoD.

Price – the cost of the billed service. Monthly price of offer or movie price for VOD. May be negative for corrective CDR.

Currency Code – identification of currency the payment is made in.

Offer ID – platform identifier of the billed offer; for OFFER CDRs only.

Offer code – identifier of the billed offer in the provider's CRM; for OFFER CDRs only.

SVoD Package ID – platform identifier of used SVoD package; for SVoD CDRs only.

SVoD Package code – identifier of the used SVoD package in the provider's CRM; for SVoD CDRs only.

TVoD category ID – platform identifier of used TVoD category; for TVoD CDRs only.

TVoD category code – identifier of used TVoD category in the provider's CRM; for TVoD CDRs only.

Reffered CDR ID – identifier of the reffered CDR, used for corrective CDRs. Note: currently there is no implemented interface allowing corrective data input. We are waiting for customers demanding this feature.

Prefix code – identifier of the purchased movie; for VOD CDRs only.

Movie name – original title of the purchased movie; for VOD CDRs only.

2.3 CDR output format

CDR output can be generated in two formats: CSV and XML. This is a Isp-specific setting and can be modified in Admin zone, the default format is CSV. The path for generated files can be set by `billing.billing-cdr-export.directory` and
`billing.content-provider-cdr-export.directory` Global config properties.

Exported files are named as: `cdr-`*`type-year-month_utcDate.ext`*`.` Type could be `tvod` or `offer`. File extension is `xml` or `csv`. File name examples:
`cdr-offer-2008-06_20080603150000Z.csv`
`cdr-tvod-2008-06_20080603150000Z.csv`

They are exported every hour and so they could contain just their header with zero records.

2.3.1 XML output format

The example output format for all types of CDR data is shown below. Price values are rounded to 6 decimal digits.

```
<?xml version="1.0" encoding="UTF-8"?>
<cdrs>
  <!-- VOD - TVoD -->
  <cdr id="26280" generated="2006-12-01T12:20:24" type="VOD" subtype=
     "TVOD">
    <subscriber id="359" code="SUB359"/>
    <subscription id="950" code="SUT950"/>
    <detail>
      <category id="25" code="TCAT25"/>
      <units>1</units>
    </detail>
    <timestamps>
      <start>2006-12-01T12:20:24</start>
      <end>2006-12-02T12:20:24</end>
    </timestamps>
    <price>
      <value>70.000000</value>
      <code>CZK</code>
    </price>
    <movie>
      <name>Gladiator</name>
      <prefix-code>VOD_TCAT25_Gladiator</prefix-code>
    </movie>
  </cdr>
  <!-- VOD - SVoD -->
  <cdr id="26281" generated="2006-12-01T12:20:24" type="VOD" subtype=
     "SVOD">
    <subscriber id="359" code="SUB359"/>
    <subscription id="950" code="SUT950"/>
    <detail>
      <tariff id="8" code="TRF8"/>
      <package id="20" code="SVOD20"/>
    </detail>
    <timestamps>
      <start>2006-12-01T12:20:24</start>
      <end>2006-12-02T12:20:24</end>
    </timestamps>
    <price>
      <value>0.000000</value>
      <code>CZK</code>
    </price>
    <movie>
      <name>Gladiator</name>
      <prefix-code>VOD_TCAT25_Gladiator</prefix-code>
    </movie>
```

```
  </cdr>
  <!-- Offer -->
  <cdr id="27173" generated="2006-12-02T01:05:02" type="OFFER">
    <subscriber id="563" code="SUB563"/>
    <subscription id="1203" code="SUT1203"/>
    <detail>
      <tariff id="8" code="TRF8"/>
      <offer id="10" code="OFF10"/>
    </detail>
    <timestamps>
      <start>2006-11-22T10:00:00</start>
      <end>2006-11-30T13:00:01</end>
    </timestamps>
    <price>
      <value>400.000000</value>
      <code>CZK</code>
    </price>
  </cdr>
</cdrs>
```

2.3.2 CSV output format

The example output format for three types of CDR data is shown below. Price values are again rounded to 6 decimal digits. The file encoding is UTF-8. The field separator is the comma character, field values are escaped with a double quote character, when needed. When the double quote character is present in the field data, it is doubled.

The CSV file has the following header:

```
id,generated,type,subtype,subscriber_id,subscriber_code, ↩
    subscription_id,subscription_code,tvod_category_id, ↩
    tvod_category_code,tariff_id,tariff_code,offer_id,offer_code, ↩
    price_value,price_code,timestamp_start,timestamp_end,units, ↩
    movie_name,movie_prefix_code
```

Example of VOD (TVoD) record (one line):

```
26280,2006-12-01T12:20:24,VOD,TVOD,359,SUB359,950,SUT950,25,TCAT25 ↩
    ,,,,,70.000000,CZK,2006-12-01T12:20:24,2006-12-02T12:20:24,1, ↩
    Gladiator,VOD_TCAT25_Gladiator
```

Example of VOD (SVoD) record (one line):

```
27173,2006-12-02T01:05:02,VOD,SVOD,563,SUB563,1203,SUT1203,,,8,TRF8 ↩
    ,20,OFF20,0.000000,CZK,2006-11-22T10:00:00,2006-11-30T13:00:01,, ↩
    Gladiator,VOD_TCAT25_Gladiator
```

Example of offer recored (one line):

```
27173,2006-12-02T01:05:02,OFFER,,563,SUB563,1203,SUT1203,,,8,TRF8,10, ↩
    OFF10,40.000000,CZK,2006-11-22T10:00:00,2006-11-30T13:00:01,,,
```

2.4 Exporting dates

New CDRs are exported every hour. VoD CDRs are produced on a movie purchase. Offer fee CDRs are produced at night between the 1st and the 2st day of a month.

2.5 Limiting subscription charges

The volume of TVoD services consumed by a subscription determines total charged costs. With prepaid service, costs are limited by the prepaid amount. With postpaid service, the debt should have a limit based on customer's payment reliability.

nangu.TV platform offers a mechanism for limiting charges connected with subscription in both prepaid and postpaid models.

2.5.1 Limiting with offline transactions

nangu.TV platform keeps for each subscription the amount of money allowed to be spent. The platform refuses to make a TVoD program purchase if the allowed amount is lower than the price of the program. The limit amount is expected to be updated by the provider's CRM via the provisioning interface.

2.5.2 Limiting with online transactions

When using online transactions, charge requests are invoked immediately on the provider's CRM TVoD webservice. If the charge request is refused, the service delivery is refused too.

The URL of the external CRM TVoD webservice is configured inside System config on Admin Zone. It is used only when it is not empty.

Purchase TVoD Entity in CRM

This call tells external CRM to charge entity purchase by an subscription. External system is expected to implement this contract:

Inputs: Data passed towards CRM System

- Subscription's code in provider's CRM.
- TVoD category code in provider's CRM.
- Entity unique id.
- Entity original title.
- Entity content provider code in provider's CRM.
- Entity price in subscription currency.

Outputs: Possible data returned by CRM System

- chargeLimit – updated subscription's charge limit in subscription currency. When it is not present, actual subscription's charge limit is decreased by the entity price.
- CrmInsufficientCreditException – Subscription does not have enough credit to purchase the entity.
- CrmNotFoundException – Subscription or the entity doesn't exist.

WSDL CRM TVoD management

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://itonis.net/tve/schema/crm/ ↩
   tvod/ns" xmlns:tns="http://itonis.net/tve/schema/crm/tvod/ns"  ↩
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"  ↩
   xmlns:soap12="http://www.w3.org/2003/05/soap-envelope" xmlns:ns1=" ↩
   http://itonis.net/tve/schema/crm/tvod" xmlns:xsd="http://www.w3. ↩
   org/2001/XMLSchema" xmlns:soapenc11="http://schemas.xmlsoap.org/ ↩
   soap/encoding/" xmlns:soapenc12="http://www.w3.org/2003/05/soap- ↩
   encoding" xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/" ↩
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns=" ↩
   http://itonis.net/tve/schema/crm/tvod" elementFormDefault=" ↩
   unqualified" targetNamespace="http://itonis.net/tve/schema/crm/ ↩
   tvod">

    <xsd:simpleType name="Code">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="32"/>
        </xsd:restriction>
    </xsd:simpleType>

     <xsd:complexType name="FaultDetail">
        <xsd:sequence>
            <xsd:element name="message" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="crmPurchaseEntityRequest">
        <xsd:annotation>
            <xsd:documentation>Tells the CRM system to charge  ↩
               purchase of the given movie entity.</xsd:documentation ↩
               >
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="subscriptionCode" type="tns:Code"/ ↩
                   >
```

```
                <xsd:element name="tvodCategoryCode" type="tns:Code"/ ↩
                    >
                <xsd:element name="entity">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="entityId" type=" ↩
                                xsd:int"/>
                            <xsd:element name="originalTitle">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string ↩
                                        ">
                                        <xsd:maxLength value="255"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="contentProviderCode"  ↩
                                type="tns:Code"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="price">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="value" type=" ↩
                                xsd:double"/>
                            <xsd:element name="currency">
                                <xsd:annotation>
                                    <xsd:documentation>Subscription  ↩
                                        currency as ISO 4217 currency  ↩
                                        code.</xsd:documentation>
                                </xsd:annotation>
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string ↩
                                        ">
                                        <xsd:length value="3"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="crmPurchaseEntityResponse">
        <xsd:annotation>
            <xsd:documentation>Returns changed charge limit.</ ↩
                xsd:documentation>
```

```
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" name="chargeLimit" type=" ↩
                    xsd:double">
                    <xsd:annotation>
                        <xsd:documentation>Limit in subscription  ↩
                            currency. When it is not present, actual  ↩
                            subscription chargeLimit is decreased by  ↩
                            the entity price.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"  ↩
   attributeFormDefault="qualified" elementFormDefault="qualified"  ↩
   targetNamespace="http://itonis.net/tve/schema/crm/tvod/ns">
<xsd:element name="crmNotFoundException" type="ns1:FaultDetail"/>
<xsd:element name="crmInsufficientCreditException" type=" ↩
   ns1:FaultDetail"/>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="crmInsufficientCreditException">
    <wsdl:part name="crmInsufficientCreditException" element=" ↩
       tns:crmInsufficientCreditException"/>
  </wsdl:message>
  <wsdl:message name="crmNotFoundException">
    <wsdl:part name="crmNotFoundException" element=" ↩
       tns:crmNotFoundException"/>
  </wsdl:message>
  <wsdl:message name="crmPurchaseEntityRequest">
    <wsdl:part name="crmPurchaseEntityRequest" element=" ↩
       ns1:crmPurchaseEntityRequest"/>
  </wsdl:message>
  <wsdl:message name="crmPurchaseEntityResponse">
    <wsdl:part name="crmPurchaseEntityResponse" element=" ↩
       ns1:crmPurchaseEntityResponse"/>
  </wsdl:message>
  <wsdl:portType name="CrmTvodEndpointPortType">
    <wsdl:operation name="crmPurchaseEntity">
      <wsdl:input name="crmPurchaseEntityRequest" message=" ↩
         tns:crmPurchaseEntityRequest"/>
      <wsdl:output name="crmPurchaseEntityResponse" message=" ↩
         tns:crmPurchaseEntityResponse"/>
      <wsdl:fault name="crmNotFoundException" message=" ↩
         tns:crmNotFoundException"/>
```

```
      <wsdl:fault name="crmInsufficientCreditException" message=" ↩
         tns:crmInsufficientCreditException"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CrmTvodEndpointHttpBinding" type=" ↩
     tns:CrmTvodEndpointPortType">
    <wsdlsoap:binding style="document" transport="http://schemas. ↩
       xmlsoap.org/soap/http"/>
    <wsdl:operation name="crmPurchaseEntity">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="crmPurchaseEntityRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="crmPurchaseEntityResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="crmNotFoundException">
        <wsdlsoap:fault name="crmNotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="crmInsufficientCreditException">
        <wsdlsoap:fault name="crmInsufficientCreditException" use=" ↩
           literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CrmTvodEndpoint">
    <wsdl:port name="CrmTvodEndpointHttpPort" binding=" ↩
       tns:CrmTvodEndpointHttpBinding">
      <wsdlsoap:address location="http://localhost:9090/tvod/ ↩
         CrmTvodEndpoint"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Chapter 3

VoD Entities Export Interface

VoD metadata could be provided to the customer (ISP) for further usage.

VoD metadata are accessible on these URLs:

- `/adminws/metadata/entities/`
- `/adminws/metadata/entities/<entityId>`
- `/adminws/metadata/media/<mediaId>.jpg`

Only normal HTTP GET is required to get the data from these URLs. Exported format is described in following sections.

3.1 Entities

Data on URL `/adminws/metadata/entities/` provide list of all released entities. The data are formated as an Atom feed. The Atom format is defined by RFC 4287[**?**].

Entities export example:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>VoD Entities</title>
  <link rel="self" href="/adminws/metadata/entities/"/>
  <updated>2008-04-23T14:33:42Z</updated>
  <author>
    <name>VoD Export</name>
  </author>
  <id>tag:iptv@alefnula.com,2008:metadata/entities/</id>

  <entry>
    <title>Elephants Dream</title>
    <link href="/adminws/metadata/entities/168"/>
    <id>tag:iptv@alefnula.com,2008:metadata/entities/168</id>
    <updated>2007-10-16T09:04:48Z</updated>
  </entry>
```

```
  <entry>
    <title>Fearless</title>
    <link href="/adminws/metadata/entities/296"/>
    <id>tag:iptv@alefnula.com,2008:metadata/entities/296</id>
    <updated>2008-04-14T09:16:17Z</updated>
  </entry>
</feed>
```

Each entity is mentioned by these fields:

title Entity original title.

link Link to entity metadata export.

id Globally unique ID.

updated Date of the last change on the entity.

3.2 Entity Metadata Export

Entity metadata are available on URL `/adminws/metadata/entities/`. CableLabs VOD Metadata format is used for entity description. The format is specified in two documents:

ADI 2.0 Specification Asset Structure – Specification of generic metadata and content holding documents.

Video-On-Demand Content Specification Version 2.0 – Specification of VoD specific metadata elements.

Visit http://www.cablelabs.com/specifications/md20.html to get the specifications.

Metadata export example:

```
<?xml version="1.0" encoding="utf-8"?>
<adi:ADI2 xmlns:vod="http://www.cablelabs.com/2006-05-05/VOD2"  ↩
   xmlns:adi="http://www.cablelabs.com/2006-05-05/ADI2" xmlns:xml=" ↩
   http://www.w3.org/XML/1998/namespace" sender="alefnula.com"  ↩
   docNumber="1" createDateTime="2008-04-23T15:35:28Z"  ↩
   relativePriority="1">
    <adi:Contact phone="+420 225 090 480" email="iptv@alefnula.com"/>
    <adi:OpenGroupAsset type="VODRelease" product="VOD">
        <vod:VODRelease providerID="HBO-code" assetID=' ↩
           ETID0000000000000168' updateNum="1" groupAsset="Y">
            <adi:AssetLifetime startDateTime="2000-01-01T00:00:00Z"  ↩
               endDateTime="3000-01-01T00:00:00Z"/>
        </vod:VODRelease>
    </adi:OpenGroupAsset>
    <adi:AddMetadataAsset type="Title" product="VOD" groupProviderID= ↩
       "HBO-code" groupAssetID='ETID0000000000000168'>
        <vod:Title providerID="HBO-code" assetID=' ↩
           TITL0000000000000168' updateNum="1">
```

```
<adi:AssetLifetime startDateTime="2000-01-01T00:00:00Z"
   endDateTime="3000-01-01T00:00:00Z"/>
<vod:Provider>HBO-code</vod:Provider>
<vod:TitleFull xml:lang="en">Elephants Dream</
   vod:TitleFull>
<vod:TitleFull xml:lang="cs">Elephants Dream</
   vod:TitleFull>
<vod:TitleBrief xml:lang="en">Elephants Dream</
   vod:TitleBrief>
<vod:SummaryLong xml:lang="en">The two main characters
   are on a journey in the folds of a giant Machine,
   exploring the twisted and dark complex of wires, gears
    and cogs. Until one moment a conflict arises that
   throws out all their assumptions. This movie short
   couples lively fun with passionate characters in an
   epic story line.</vod:SummaryLong><vod:SummaryLong
   xml:lang="cs">The two main characters are on a journey
    in the folds of a giant Machine, exploring the
   twisted and dark complex of wires, gears and cogs.
   Until one moment a conflict arises that throws out all
    their assumptions. This movie short couples lively
   fun with passionate characters in an epic story line.<
   /vod:SummaryLong>
<vod:SummaryMedium xml:lang="en">The two main characters
   are on a journey in the folds of a giant Machine,
   exploring the twisted and dark complex of wires, gears
    and cogs. Until one moment a conflict arises that
   throws out all their assumptions. This movie short
   couples lively fun with pa</vod:SummaryMedium>< 
   vod:SummaryMedium xml:lang="cs">The two main
   characters are on a journey in the folds of a giant
   Machine, exploring the twisted and dark complex of
   wires, gears and cogs. Until one moment a conflict
   arises that throws out all their assumptions. This
   movie short couples lively fun with pa</
   vod:SummaryMedium>
<vod:SummaryShort xml:lang="en">The two main characters
   are on a journey in the folds of a giant</
   vod:SummaryShort><vod:SummaryShort xml:lang="cs">The
   two main characters are on a journey in the folds of a
    giant</vod:SummaryShort>
<vod:RunTime>634</vod:RunTime>
<vod:DisplayRunTime>00:10</vod:DisplayRunTime>
<vod:Year>2006</vod:Year>
<vod:CountryOfOrigin>US</vod:CountryOfOrigin>

<vod:Actor>
    <vod:LastNameFirst>Fialová, Květa</vod:LastNameFirst>
    <vod:LastName>Fialová</vod:LastName>
```

```
            <vod:FirstName>Květa</vod:FirstName>
        </vod:Actor>
        <vod:Actor>
            <vod:LastNameFirst>Jetenská, Kristýna</ ↩
               vod:LastNameFirst>
            <vod:LastName>Jetenská</vod:LastName>
            <vod:FirstName>Kristýna</vod:FirstName>
        </vod:Actor>

        <vod:Director>
            <vod:LastNameFirst>Macharáček, Ivo</vod:LastNameFirst ↩
               >
            <vod:LastName>Macharáček</vod:LastName>
            <vod:FirstName>Ivo</vod:FirstName>
        </vod:Director>
        <vod:Director>
            <vod:LastNameFirst>Žatečka, Tomáš</vod:LastNameFirst>
            <vod:LastName>Žatečka</vod:LastName>
            <vod:FirstName>Tomáš</vod:FirstName>
        </vod:Director>

        <vod:Genre xml:lang="en">Adult</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Pro dospělé</vod:Genre>
        <vod:Genre xml:lang="en">Horror</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Horor</vod:Genre>
        <vod:Genre xml:lang="en">Drama</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Drama</vod:Genre>
        <vod:Genre xml:lang="en">Comedy</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Komedie</vod:Genre>
        <vod:Genre xml:lang="en">Crime</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Krimi</vod:Genre>
        <vod:Genre xml:lang="en">Adven</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Dobrodružný</vod:Genre>
        <vod:Genre xml:lang="en">Documentary</vod:Genre>< ↩
           vod:Genre xml:lang="cs">Dokumentární</vod:Genre>
        <vod:Genre xml:lang="en">Fiction</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Fikce</vod:Genre>
        <vod:Genre xml:lang="en">Action</vod:Genre><vod:Genre  ↩
           xml:lang="cs">Akční</vod:Genre>
    </vod:Title>
    <vod:CategoryPath providerID="HBO-code" assetID=' ↩
       ETID0000000000000168' updateNum="1" mso ="">
        <adi:AssetLifetime startDateTime="2000-01-01T00:00:00Z"  ↩
           endDateTime="3000-01-01T00:00:00Z"/>
        <vod:Path>Movie</vod:Path>
        <vod:CategoryPrice format="SD" currency="CZK">59.0</ ↩
           vod:CategoryPrice>
        <vod:CategoryPrice format="SD" currency="EUR">2.49</ ↩
           vod:CategoryPrice>
```

```
        </vod:CategoryPath>
    </adi:AddMetadataAsset>
    <adi:AddMetadataAsset type="Poster" product="VOD" groupProviderID ↩
       ="HBO-code" groupAssetID='ETID0000000000000168'>
        <vod:Poster providerID="HBO-code" assetID=' ↩
           POST0000000000000168' updateNum="1">
            <adi:AssetLifetime startDateTime="2000-01-01T00:00:00Z"  ↩
               endDateTime="3000-01-01T00:00:00Z"/>
            <vod:PosterType type="standard">
                <vod:ContentRef providerID="HBO-code" assetID=' ↩
                   MEDI0000000000019194' updateNum="1"/>
            </vod:PosterType>
        </vod:Poster>
    </adi:AddMetadataAsset>
    <adi:AcceptContentAsset type="Image" fileName="19194.jpg"  ↩
       fileSize="637701" mD5CheckSum="1 ↩
       b57149e9d002c4bcceddb89f163ec93" transferContentURL="/adminws/ ↩
       metadata/media/19194.jpg">
        <vod:Image providerID="HBO-code" assetID=' ↩
           MEDI0000000000019194' updateNum="1" fileName="19194.jpg"  ↩
           fileSize="637701" mD5CheckSum="1 ↩
           b57149e9d002c4bcceddb89f163ec93" imageEncodingProfile=" ↩
           OTHER" transferContentURL="/adminws/metadata/media/19194. ↩
           jpg">
            <adi:AssetLifetime startDateTime="2000-01-01T00:00:00Z"  ↩
               endDateTime="3000-01-01T00:00:00Z"/>
            <vod:ImagePixels horizontalPixels='1468' verticalPixels=' ↩
               2133'>.jpg</vod:ImagePixels>
        </vod:Image>
    </adi:AcceptContentAsset>
</adi:ADI2>
```

3.2.1 Changes to the specification

- Text lenghts are not limited when the information does not fit in any larger field. For example, SummaryMedium maximal length is respected, but SummaryLong is used as unlimited.
- RunTime type is number of seconds, it is not "hh:mm:ss".
- Number of Director, Actor and Genre elements is unlimited.
- LastNameFirst is not limited to ASCII.

3.2.2 Notes on XML processing

- The Summary* elements could contain text with escaped HTML.

- The CableLabs XML Schema for VoD documents is incomplete. The specification is the authoritative source.

3.3 Media Export

An entity cover image referenced from the metadata is available on URL `/adminws/metadata/media/<mediaId>.jpg`.

Chapter 4

Mosaic Encoder Control Interface

Communication between the Application Server and Mosaic Encoder is bi-directional. The encoder connects to the AS to download its configuration, while the AS can connect to the encoder to notify it about the configuration changes.

The encoder retrieves the configuration XML data by a HTTP GET request to `/adminws/config/mosaic/hostname:port`, where hostname:port is the address and port of encoder's HTTP server for backward notifications.

When the encoder runs, a GET request at `http://hostname:port/restart` forces the encoder to restart and reload its configuration.

Mosaic configuration consists of the following elements:

- `output` — describes the streaming parameters
  - `address` — destination IP address
  - `port` — destination port
  - `bitrate` — streaming bitrate in bps
- `dimensions` — target dimensions in pixels
  - `w` — width in pixels
  - `h` — height in pixels
  - `darnum` — Numerator of the aspect ratio
  - `darden` — Denominator of the aspect ratio
- `screen` — description of one mosaic screen
  - `uri` — source URI or a list of URIs separated by semicolons
  - `x` — horizontal position
  - `y` — vertical position
  - `w` — width
  - `h` — height

- `overscan` — overscan percentage
- `interlaced` — true if scaling both top and bottom fields, false if scaling the top field only
- `language` — preferred language for the screen

Example configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE mosaic SYSTEM "file:///etc/tve/mosaic.dtd">
<mosaic>
<output address='239.194.100.66' port='1234' bitrate='8000000'/>
<dimensions w='720' h='576'/>
<screen uri='udp://239.2.3.100:2314'
        x='60' y='208'
        w='200' h='160'
        overscan='10' interlaced='false'
        language=''/>
<screen uri='udp://239.2.3.100:2314'
        x='260' y='208'
        w='200' h='160'
        overscan='10' interlaced='false'
        language=''/>
<screen uri='udp://239.2.3.100:2314'
        x='460' y='208'
        w='200' h='160'
        overscan='10' interlaced='false'
        language=''/>
</mosaic>
```

Chapter 5

Operator PVR

5.1 Introduction

ISP may record the selected programs according epg parameters. Selection of programs for recording is accomplished by web services. After a program is successfully recorded, it behaves like VOD entity. Web services also provide an interface for monitoring recorded programs. Entity mentioned above are called Operator PVR (OPVR).

5.2 Recording of programs

The recording program interface listens on: `http://<server>:<port>/adminws/opvr/rpc`. It expects communication via XML-RPC and handles one function: `recordProgram`. The function has the following parameters:

- `name` – The name of the program.
- `language` – The language in which the name of the program is entered.
- `startDate` – The start date of the program (date format is defined by ISO 8601 (YYYYMMDD'T'HH:mm:ss).
- `endDate` – The End date of the program.
- `channelKey` – The channel key.

If a program is successfully scheduled for recording, it sends the `opvr_program_id` as a response, otherwise a error response with one of the following error codes is sent:

- `1` – Invalid request parameters.
- `2` – Channel not found for the given channel key.
- `3` – EPG not found according given parameters.
- `4` – Program was found, but it can't be recorded. (e.g. program has ended and record of such programs are forbidden)

5.3 Monitoring of recording programs

Is possible to monitor the recording programs in different states on the following URLs:

- `http://<server>:<port>/adminws/opvr/export/recordedPrograms`, parameters:
  - `startDate` – Minimum timestamp of entity creation. (optional) (Unix time)
  - `endDate` – Maximal timestamp of entity creation. (optional)
- `http://<server>:<port>/adminws/opvr/export/scheduledPrograms`, parameters:
  - `startDate` – Minimum timestamp of the start of the EPG. (optional)
  - `endDate` – Maximal timestamp of the end of the EPG. (optional)
- `http://<server>:<port>/adminws/opvr/export/recordingPrograms`

Only normal HTTP GET is required to get the data from these URLs. Because the recorded programs behave like VOD entities, the response to the request for recorded programs is in a different format than the other. The first one conforms ATOM format (RFC 4287) and is described in section 3, the other two conforms to JSON (RFC 4627). In JSON format are exported EPGs selected for recroding and their record is not over.

Example of export of the scheduled programs for recording in JSON format:


```
{
  "CT2": [
    {
      "epgId": 3956435,
      "channelKey": "CT2",
      "entityId": null,
      "startTimestamp": 1257459900,
      "endTimestamp": 1257464400,
      "npvrEnabled": null,
      "timeShiftEnabled": null,
      "picture": "/media/MOVIE_DETAIL/379716.jpg",
      "name": "Persepolis"
    },
    {
      "epgId": 3956434,
      "channelKey": "CT2",
      "entityId": null,
      "startTimestamp": 1257459600,
      "endTimestamp": 1257459900,
      "npvrEnabled": null,
      "timeShiftEnabled": null,
      "picture": null,
      "name": "Minuty z Mezipater"
    }
  ],
  "Nova": [{
```

```
    "epgId": 3954841,
    "channelKey": "Nova",
    "entityId": null,
    "startTimestamp": 1257444000,
    "endTimestamp": 1257446100,
    "npvrEnabled": null,
    "timeShiftEnabled": null,
    "picture": null,
    "name": "Udalosti"
  }]
}
```


5.4 Configuration parameters

Some parameters of newly created OPVR entity can be set in the global config. The particular parameters are:

- `opvr.content-provider.prefix` – The Content provider name is created as `prefix + channelKey`. (`opvr-` by default)
- `opvr.price-level.name` – The name of price level. (default `opvr` by default)
- `opvr.licensing-window.duration` – The duration of newly created licensing window for OPVR entities in hours. (10 years by default)

References

[1] Martin Cizek, Michal Vojtisek, Antonin Kral, and Ivo Danihelka, "Billing interface," Tech. Rep. ITNS-BILL, ITonis Inc.