Tuner Reservation Manager Specification

Work In Progress

Version 1.0.8

Document Status

Document Control Number:	
Document Title:	Tuner Reservation Manager Specification
Versions:	Initial version May 23, 2013
	Revisions: - Jun12, 2014: Update per RDK-3674 - Feb 1, 2014: Update per XREINT-1000. Aug 12 2013: Added Section 2.4 to include utility TRM messages. – Aug 6 2013: correct lower/ upper case spelling in messages.– July 31 2013: Incremental changes made along with TRM meetings and discussions. – May 31 2013: Added more details and sequence diagrams to address questions raised from the 05/ 30 meeting. – May 29 2013: updated messages to match rules and current implementation. Added Rules section.
Date:	May 2013
Status:	Work In Progress
Distribution:	Comcast and External parties under NDA

Document Status Codes

Work in Progress (WIP)	An incomplete document designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
Draft (D)	A document in specification format considered largely complete, but lacking review. Drafts are susceptible to substantial change during the review process.
Issued (I)	A stable document which has undergone rigorous review and is suitable for product design and development. It will serve as a basis for testing requirements.

Table of Contents

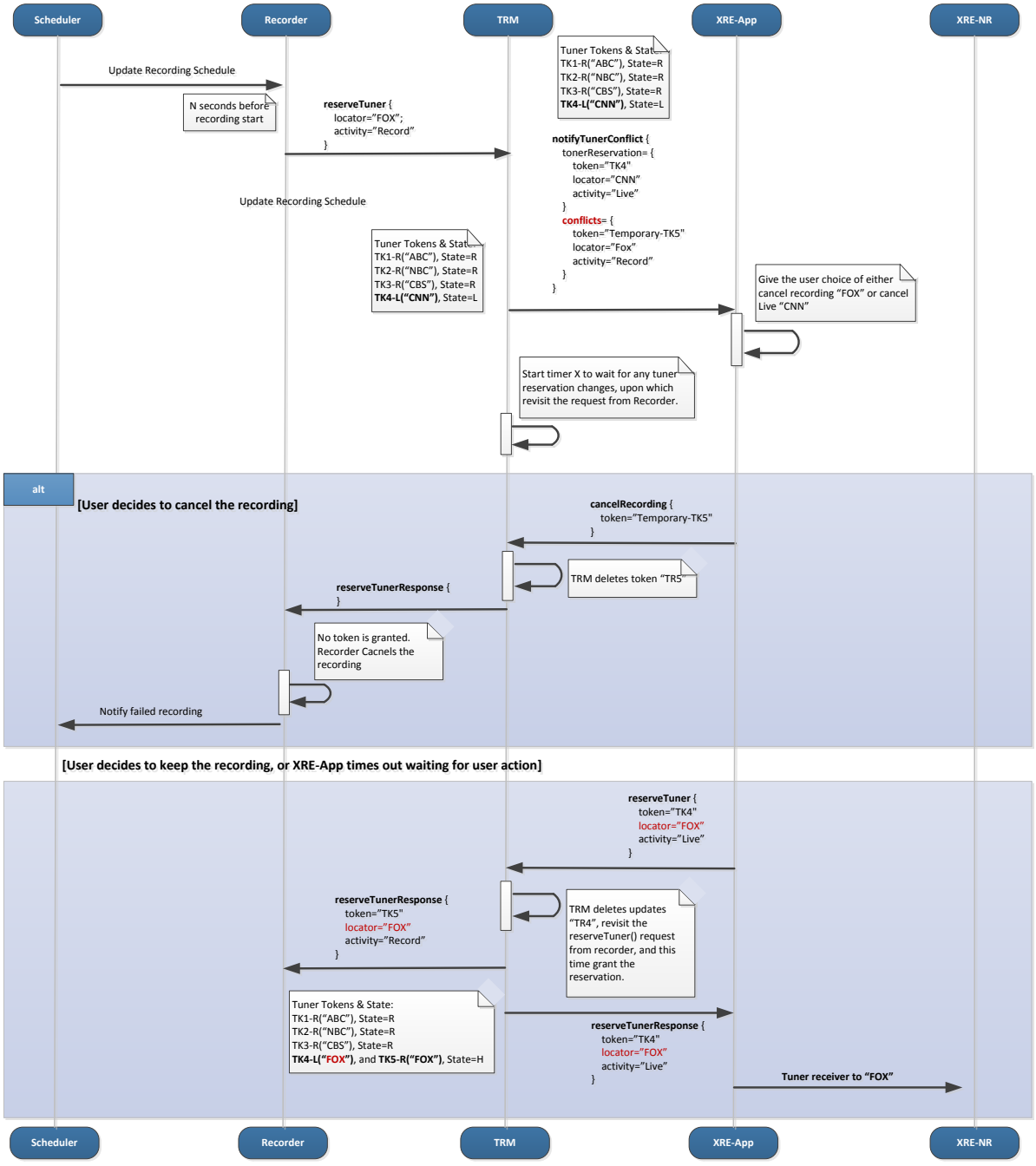
Section 1: Overview

Tuner Reservation Manager (TRM) runs on a gateway device and coordinates the usage of tuners on the device for its connected clients.

1.1 Sequence Diagrams

1.1.1 IP Client Recording start with conflict

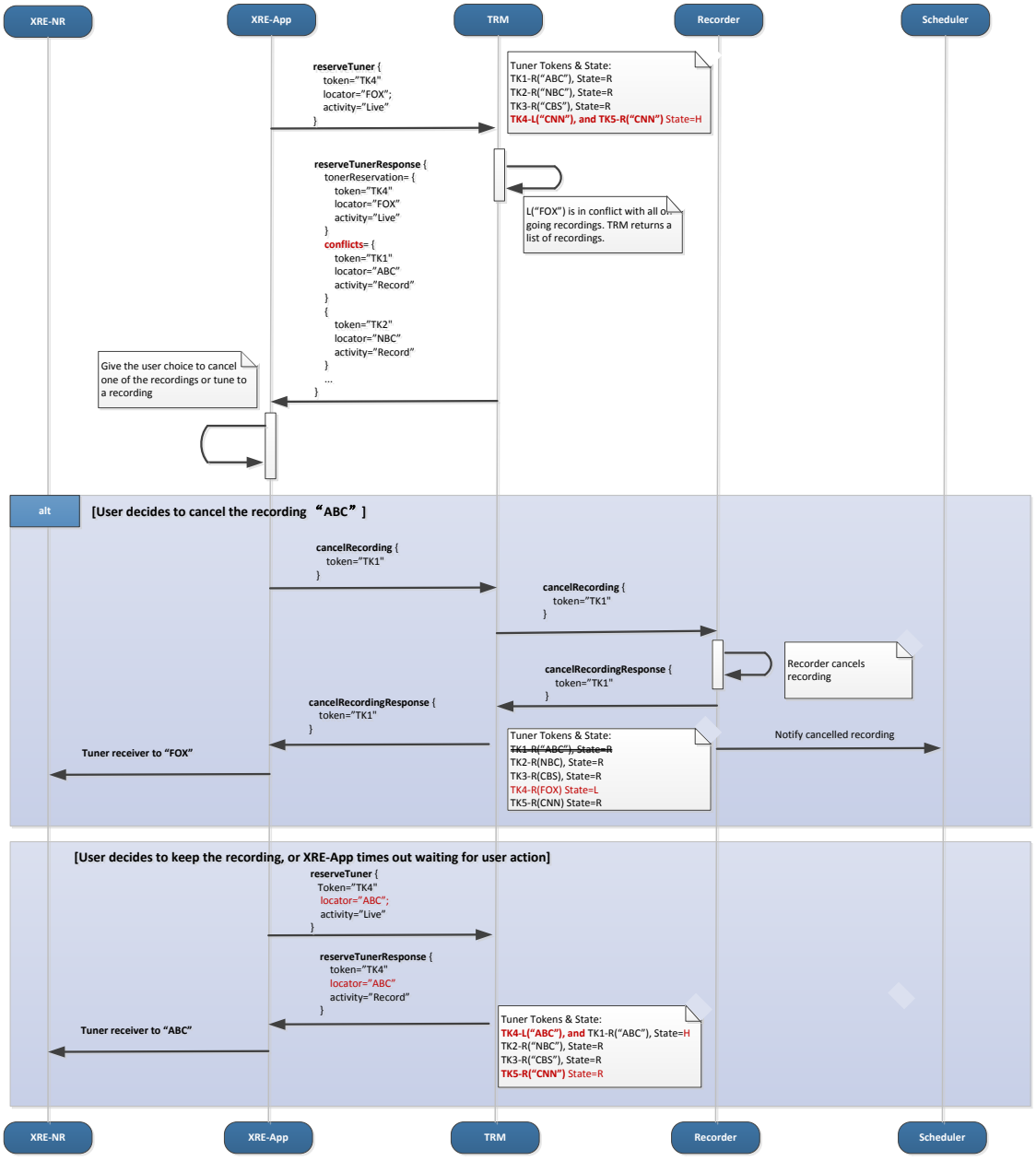
In this sequence diagram, 3 tuners are recording (“ABC”, “NBC”, “CBS”) and user is watching CNN. Scheduled Recording of “FOX” is about to start.



XI-3 Recording start with Conflict

1.1.2 IP Client Channel Change Conflicts with Recording

In this sequence diagram, 3 tuners are recording ("ABC", "NBC", "CBS") and user is watching and recording CNN on 4th tuner. Now user tries to tune to "FOX". Since user already owns an L reservation for watching CNN, it can reuse the same token. This L-Reservation is moved from tuner to tuner, toggling the states of affected tuners between L and H.



XI-3 Channel change conflicts with Recording

Section 2: Message Specification

2.1 Common Entities

Certain information will be shared among messages, as described below.

2.1.2 CustomAttributes

Tuner State object represents state of the tuner.

```
CustomAttributes :=
{
  /* JSON Entities defined by application */
}
```

The **CustomAttribute** contains JSON Entities defined by the application. It is sent by the application when it requests for a tuner reservation. After the reservation is granted, and as long as the content of the reservation is not modified, the customAttributes will be present in any messages, including the asynchronous notifications, that contains the corresponding TunerRevervation. If a renewed TunerReservation also contains **CustomAttribute**, the new attributes object will take the place. See more on this in NotifyTunerReservationUpdate message.

The lifetime of the object is then same as the lifetime of the TunerReversation itself.

2.1.3 Activity

Activity object represents the request or granted usage of a tuner.

```
Activity :=
{
  "name"              : [String] name,
  "details"(optional): <Details>,
}
```

The **activity** field represents the intended use of the tuner. Supported tuner activity names are:

- **Live**: the tuner is used for *Live* (Live Streaming or Local Live).
- **Record**: the tuner is used for *Recording*.
- **EAS**: the tuner is used for *EAS*.

When tuner sharing is allowed among these activities, a tuner may be reserved for multiple activities at a time. This is indicated by the **state** of the tuner. However, a

single Tuner Reservation Request message can contain at most one **activity**. The **details** of **activity** is optional.

2.1.4 Details

Each **activity** may be associated with a set of **details** describing the activity.

```
Details :=
{
  "recordingId"             : [String]
  "hot"                     : [String]
}
```

The fields specified here are required for the associated **activity**. The requestor is allowed to insert unspecified fields in the **details**. These unspecified fields are ignored by TRM, and echoed back in response messages that have the **activity** field. The defined fields are:

- **recordingId**: required when requesting **Record** activity for a tuner.
- **hot** : flag (true or false) indicating of the recording is scheduled or hot.

2.1.5 State

Tuner State object represents state of the tuner.

```
State :=
{
  "state"             : [String] state,
}
```

The **state** field indicates the activity state of a tuner:

- **Live**: the tuner is reserved for Live activity (Streaming or Playback).
- **Record**: the tuner is reserved for Record activity.
- **Hybrid**: the tuner is reserved for Live **and** Record activity.
- **EAS**: the tuner is reserved for EAS.
- **Free**: the Tuner is not reserved.

2.1.6 DetailedTunerState

```
DetailedTunerState :=
{
    "state" : <State>
    "serviceLocator" : [String] sourceLocator
    "owners" : {
        <Activity> : {
            "device" : [String] device
        }
        ...
    }
    "reservedDeviceId" : [String]
}
```

- **reservedDeviceId**: If this value is present in the **detailedStates**, the corresponding tuner is can only be used for the following usages.
    - **LIVE** streaming for the device represented by the **reservedDeviceId**.
    - **RECORD** for any recording request, regardless of the recording's originating device.

2.1.7 AllTunerState

```
AllTunerStates :=
{
  <tunerId>          : <State>
   ...
}
```

2.1.8 AllTunerDetailedStates

```
AllTunerDetailedStates :=
{
  <tunerId>          : <DetailedTunerState>
   ...
}
```

2.1.9 TunerReservation

The TunerReservation object represents a requested or granted tuner reservation. The reservation has a validity window that is indicated by **startTime** and **duration**. The requesting device is required to renew a reservation before its validity window disappears. If it is not renewed, the token will be released by TRM and all messages that follow bearing the token will be considered as **MalformedRequest**. For Record reservations, the requested **startTime** should be **N** seconds ahead of the actual recording start time (or should be left out so that the granted reservation starts at the time when it is granted by TRM), to give room in case a conflict needs to be resolved.

A same **reservationToken** can be reused if and only if values of {**device, activity**} are the same. In the case, the {**serviceLocator, startTime, duration**} of the reused token can be updated. This is useful during a channel change, where the TRM client can reuse a same Live tuner reservation by just updating the **serviceLocator** of it.

A reservation is renewed by requesting **tunerRreservation** with a same **reservationToken**.

```
TunerReservation :=
{
  "reservationToken" (optional) : [String] reservationToken,
  "device"                      : [String] device,
  "serviceLocator"              : [String] sourceLocator,
  "startTime" (optional)        : [long]   startTime,
  "duration"  (optional)        : [long]   duration
  "activity"                    : <Activity>
  "customAttributes" (optional) : <CustomAttributes>

}
```

- **token**: a unique security token generated when a reservation is created. After a reservation is created, this token is used in all messages to uniquely identify an existing reservation within TRM.
- **device:** the remote device requesting the reservation. For a hot recording, this
-
- should be the receiverId of the originating device.
- **serviceLocator:** locator of the service that the tuner will tune to.
- **startTime:** start time of the reservation in milliseconds from the epoch. If not present in a request message, this is set to when the reservation is granted or renewed. **startTime** is always included in a response message.
- **duration:** the reservation period measured from the start in milliseconds. If not present in a request message, the token is valid for a default duration. **duration** is always included in a response message.
- **activity:** the granted **activity**. Granted **activity** may or may not be the same as the requested. In the latter case the owner of the reservation will need

to comply with the returned **activity**, or initiate conflict resolution. For example, when a client requests **Live** activity when **EAS** is in progress, the returned reservation will have the **EAS** activity.

- **customAttributes:** a set of attributes assigned by the application. These attributes are associated with the reservation token for the lifetime of the reservation.

2.1.10 ResponseStatus

All response messages from TRM will provide information regarding the status of the response. Responses to recognized requests may contain additional information, as described in later sections of this document. Responses to unrecognized requests will contain only this status data, consisting of a status code and message signifying the request was unrecognized.

```
ResponseStatus := {
   "status"                      : [String] status,
   "statusMessage"   [optional]  : [String] statusMessage
}
```

Where the fields are defined as follows:

- **status**: is an enumeration of strings indicating the status of the request.
  - **Ok** Request was successful
  - **GeneralError** Request was unsuccessful
  - **MalFormedRequest** Unexpected/ Invalid request data
  - **UnRecognizedRequest** Unrecognized request
  - InsufficientResource: there is no tuner available
  - UserCancellation: Token is released as result of user cancellation.
  - InvalidToken: Token included in the message is invalid.
- **statusMessage**: is a string containing additional information about the status.

2.2 Messages

The messages will follow the request-response message exchange pattern. The general format of the request payload is:

```
{
  RequestName : {
    "requestId"                          : [String] requestId,
    "device"                             : [String] device,

  }
}
```

where the fields are defined as follows:

- **RequestName**: the name of the request, it will vary for the operations. This generally should be the device where the request message originates.
- **requestId**: a GUID used to match requests with responses. For every request, the client supplies a **requestId** that it can use to match the corresponding response. For every notification, the sender supplies a **requestId**.
- **device**: the remote device making the request.

The general format of the response payload is:

```
{
  response: {
    "requestId"                          : [String] requestId,
    <ResponseStatus>

  }
}
```

where the fields are defined as follows:

- **requestId**: matches the response to a request.

2.2.1 ReserveTuner

The client uses this message to request, update or renew a reservation. When renewing a reservation, the tunerReservation contains an existing reservationToken and matching values of {**device, activity**}

```
{
  "reserveTuner" : {
    "requestId"         : [String] requestId,
    "device"            : [String] device,
    "tunerReservation" : <TunerReservation>
  }
}
```

where the fields are defined as follows:

-
- **<tunerReservation>:** details of the requested, renewed or updated reservation. If the **tunerReservation** contains **reservationToken** entity, the request is to renew or update an existing reservation identified by the token. Only certain values of **tokenReservation** can be updated. [see 2.1.4]
- **reservationToken:** If present and valid in **<tunerReservation>**, the existing reservation will be updated. The field should not be present in the initial ReserveTuner message. The value can only be created by TRM.

2.2.2 ReserveTunerResponse

This is response message to a ReserveTuner request. The tuner reservations included in the response are granted reservations. The requesting component (e.g. XRE-App or Recorder) must comply with the granted reservations, or update the granted reservations with the actual tuner usage.

```
{
  "reserveTunerResponse"  : {
    "requestId"           : [String] requestId,
    <ResponseStatus>
    "tunerReservation"    : <TunerReservation>
    "conflicts"           : [<TunerReservation>,...]
  }
}
```

where the fields are defined as follows:

- **tunerReservation**: the returned/ granted reservation. This may or may not exist in the response message depending on the conflict resolution rules used

by TRM. If present, its values may be different (in **serviceLocator**, or in **startTime** and such) from what are requested. When this field is present and there is no **conflicts** in the response, the owner of the token must comply with the activity in the response message. When this field is present and there is **conflicts** in the response, the granted **tunerReservation** is compatible to those listed in **conflicts**. The owner of the granted token must either accept the granted resolution as is or resolve the conflict without acting on the granted reservation. After conflict is resolved, and the resolution results in different from the granted reservation, the owner must update the granted token with the new values. When this filed is not present, the owner of the token must initiate resolution and then try reservation again.

- **conflicts**: an array of existing tuner reservations that is in conflict with the requested reservation.

2.2.3 ReleaseTunerReservation

The client can release an existing tuner reservation. Normally, a token need not be released if its owner **device** and **activity** have not changed. In this case a next TunerReservation request will update the same token with new values of {**servicelocator, startTime, duration**}The format of releasing a reservation is.

```
{
  "releaseTunerReservation" : {
    "requestId"          : [String] requestId,
    "device"             : [String]   device,
    "reservationToken"   : [String] reservationToken
  }
}
```

where the fields are defined as follows:

- **reservationToken**: the reservation to be released. After successful release, the token is no longer valid and cannot be used in future messages. It is client's responsibility to stop the associated **activity** first before releasing the tuner reservation with TRM.

2.2.4 ReleaseTunerReservationResponse

The general format of the response payload is:

```
{
  releaseTunerReservationResponse: {
    "requestId"     : [String] requestId,
    <ResponseStatus>
    "reservationToken"   : [String] reservationToken
    "released"           : [String] released

  }
}
```

where the fields are defined as follows:

- **reservationToken**: the reservation released. After successful release, the token is no longer valid and cannot be used in future messages.
- **Released**: a string of "true" of "false", indicating if an valid tocken is successfully released. If the tocken is not valid, the **ResponseStatus** will carry "**MALFORMED_REQUEST**"

2.2.5 ValidateTunerReservation

The client can validate an existing tuner reservation. A reservation is valid after it is created, and invalid after it is released either per client's request or upon receiving asynchronous notification from TRM. The format of validating a reservation is:

```
{
  "validateTunerReservation" : {
    "requestId"         [required] : [String] requestId,
    "device"            [required] : [String]   device,
    "reservationToken"  [required] : [String] reservationToken
  }
}
```

where the fields are defined as follows:

- **reservationToken**: the reservation to be validated.

2.2.6 ValidateTunerReservationResponse

The general format of the response payload is:

```
{
  validateTunerReservationResponse: {
    "requestId"     : [String] requestId,
    <ResponseStatus>
    "reservationToken"   : [String] reservationToken
    "valid"              : [String] valid
  }
}
```

where the fields are defined as follows:

- **reservationToken**: the reservation validated.
- **valid**: a string of **"true"** or **"false"** indicates if the token is valid or not. The token is valid only if the reservationToken and device both match the record maintained by TRM.

2.2.7 CancelRecording

The client uses this message to request TRM to cancel the recording associated with the supplied **reservationToken**. After successReful cancelation of the recording, the associated token is no longer valid.

```
{
  "cancelRecording" : {
    "requestId"        : [String] requestId,
    "reservationToken" : [String] reservationToken,
  }
}
```

where the fields are defined as follows:

- **reservationToken:** the token whose recording is to be cancelled. The **activity** of the token must be **Record.**

2.2.8 CancelRecordingResponse

This is response message to a CancelRecording request.

```
{
  "cancelRecordingResponse" : {
    "requestId"           : [String] requestId,
    <ResponseStatus>

    "reservationToken "   : [String] reservationToken
    "cancelled"            : cancelled
  }
}
```

where the fields are defined as follows:

- reservationToken: The reservation that is cancelled.
- **state:** state of the tuner after reservation is cancelled.
- **cancelled:** if the recording is cancelled. "false" is also returned if the recording does not exist.

2.3 Notifications

Notifications are asynchronous messages sent by TRM to token owners with regard to status update of an existing reservation.

2.3.1 NotifyTunerReservationConflict

notifyTunerReservationConflict is an asynchronous notification from TRM to the owner of a token that a tuner reservation is about to be terminated, unless the owner initiates to resolve the conflict. If no conflict resolution is provided, the current reservation will be terminated automatically at the **startTime** of the new reservation. The owner of current reservation will receive **notifyTunerReservationRelease** notification when its reservation is released by TRM. If any value of the tunerReservation does not match those held by the owner, the **notifyTunerConflict** should be discarded. This can happen if user has changed tuner activity after the **notifyTunerConflict** is sent.

```
{
  "notifyTunerReservationConflict" : {
    "requestId"           : [String] requestId,
    "tunerReservation"    : <TunerReservation>
    "conflicts"           : [<TunerReservation>]
  }
}
```

where the fields are defined as follows:

- **tunerReservation**: details of the current reservation that is to be cancelled as result of a conflict. This reservation is in conflict with those listed in **conflicts**. Upon notification, the owner must initialize conflict resolution if it wishes to retain its reservation.
- **conflicts**: one or more reservations that are going to override current reservation. The reservations listed in must **conflicts** be cancelled if the owner wishes to retain the reservation in **tunerReservation.**

2.3.2 NotifyTunerReservationRelease

Asynchronous Notification from TRM to the owner of a token that its tuner reservation has been terminated. The token is no longer valid after receiving this message.

```
{
  "notifyTunerReservationRelease" : {
    "requestId"              : [String] requestId,
    "reservationToken"       : [String] reservationToken
    "reason"                 : [String] reason
  }
}
```

where the fields are defined as follows:

- **reservationToken**: the tuner reservation that is terminated.
- **reason**: a message explaining why the reservation is terminated.

2.3.3 NotifyTunerStatesUpdate

Asynchronous Notification from TRM whenever a tuner has changed it state. The TRM client can listen for this notification and keep a local cache of the tuner states. Note the payload represents the states when the message is generated. Tuner states could change after the message is sent.

```
{
  "notifyTunerStatesUpdate" : {
    "requestId"              : [String] requestId,
    "detailedState"          : <AllTunerDetailedStates>
  }
}
```

where the fields are defined as follows:

- **detailedStates**: state Information about each tuner.

2.3.4 NotifyTuneReservationUpdate

Asynchronous Notification from TRM whenever a reservation has changed it usage by its owner. The Server can listen for this notification and synchronize its cached value with the new reservation. In the update message, these fileds are guaranteed to be the original value associated with the reservation

- tunerReservation::reservationToken
- tunerReservation::device
- tunerReservation::activity
- tunerReservation::customAttributes

```
{
  "notifyTunerReservationUpdate" : {
    "requestId"              : [String] requestId,
    "tunerReservation"       : <TunerReservation>
  }
}
```

where the fields are defined as follows:

- **tunerReservation**: the reservervation that has changed.

2.4 Utility Messages

Utility Messages provide useful information about the status of the TRM.

2.4.1 GetVersion

`getVersion` lets TRM clients query server's version. The TRM server and client shall maintain backward compatibility.

```
{
  "getVersion" : {
    "requestId"    : [String] requestId,
  }
}
```

2.4.2 GetVersionResponse

```
{
    getVersionResponse: {
        "requestId" : [String] requestId,
        <ResponseStatus>
        "version" : [String] version number
    }
}
```

2.4.3 GetAllTunerIds

getAllTunerIds requests for the unique ID that system has assigned to each tuner. The ID for each tuner is guaranteed to be unique within a same target host.

```
{
  "getAllTunerIds" : {
    "requestId"    : [String] requestId,
  }
}
```

2.4.4 GetAllTunerIdsResponse

getAllTunerIdsResponse requests for the unique ID that system has assigned to each tuner. The ID for each tuner is guaranteed to be unique within a same target host.

```
{
  "getAllTunerIdsResponse" : {
    "requestId"    : [String] requestId,
    <ResponseStatus>
    "tunerIds"     : [String,...]
  }
}
```

- **tunerIds**: An array of **String**. Each element is an ID of a tuner.

2.4.5 GetAllTunerStates

getAllTunerStates requests for the **State** of all tuners on the system. Each tuner can be in any of the following state specified in 0.

```
{
  "getAllTunerStates" : {
    "requestId"           : [String] requestId,
  }
}
```

2.4.6 GetAllTunerStatesResponse

getAllTunerStatesResponse requests for all tuner reservations that are valid at the time of the request.

```
{
  "getAllTunerStatesResponse" : {
    "requestId"           : [String] requestId,
    <ResponseStatus>
    "allStates"           : <AllTunerStates>
    "detailedStates"      : <AllTunerDetailedStates>
  }
}
```

where the fields are defined as follows:

- **tunerId**: A string returned from **getAllTunerIdsResponse**.
- **AllTunerStates:** An enumeration of tuner states at time of request. This is deprecated and replaced by "AllTunerDetailedStates" since 1.0.5. For backward compatibility, the 1.05 or newer TRM client should ignore this field if it is present in the response message.
- **AllTunerDetailedStates:** An enumeration of detailed tuner states at time of request.

2.4.7 GetAllReservations

getAllReservations requests for all tuner reservations that are valid at the time of the request.

```
Filter :=
{
  "device" (optional)     : [String] device,
  "activity" (optional)   : <Activity>,
  "tunerState" (optional) : <State>,
}
```

```
{
  "getAllReservations" : {
    "requestId"           : [String] requestId,
    "filters"             : [<Filter>,...]
  }
}
```

- **filters**: An array of filtering values Each filter can contain any number of conditions. Valid condition include **device**, **Activity**, **State**. If a **filters** has multiple conditions, these conditioned are AND'd to generate the final result. If there are multiple **filters**, these filters are OR'd to generate the final result.

2.4.8 GetAllReservationsResponse

getAllReservationsResponse requests for all tuner reservations that are valid at the time of the request.

```
AllTunerReservations :=
{
    <tunerId>             : [<TunerReservation>, ...]
}
```

```
{
  "getAllReservationsResponse" : {
    "requestId"           : [String] requestId,
    "allTunerReservations": <AllTunerReservations>
    ...
  }
}
```

- **tunerId**: A string returned from **getAllTunerIdsResponse**. Each **tunerId** maps to an array of **tunerReservation** currently active on the tuner.