ETSI TR 102 397-8 V1.1.1 (2005-08)

Technical Report

Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs; Part 8: Terminal Status Mapping

Reference

DTR/TISPAN-01021-08-OSA

Keywords

API, OSA, service

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

The present document is part 8 of a multi-part deliverable covering Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs, as identified below:

Part 1: "Common Mapping";

Part 2: "Third Party Call Mapping";

Part 3: "Call Notification Mapping";

Part 4: "Short Messaging Mapping";

Part 5: "Multimedia Messaging Mapping";

Part 6: "Payment Mapping";

Part 7: "Account Management Mapping";

Part 8: "Terminal Status Mapping";

Part 9: "Terminal Location Mapping";

Part 10: "Call Handling Mapping";

Part 11: "Audio Call Mapping";

Part 12: "Multimedia Conference Mapping";

Part 14: "Presence Mapping".

NOTE: Part 13 has not been provided as there is currently no defined mapping between ES 202 391-13 [4] and the Parlay/OSA APIs. If a mapping is developed, it will become part 13 of this series.

The present document has been defined jointly between ETSI, The Parlay Group (http://www.parlay.org) and the 3GPP.

1 Scope

The present document specifies the mapping of the Parlay X Terminal Status Web Service to the Mobility User Status Service Capability Feature (SCF).

The Parlay X Web Services provide powerful yet simple, highly abstracted, imaginative, telecommunications functions that application developers and the IT community can both quickly comprehend and use to generate new, innovative applications.

The Open Service Access (OSA) specifications define an architecture that enables application developers to make use of network functionality through an open standardized interface, i.e. the Parlay/OSA APIs.

2 References

For the purposes of this Technical Reports (TR), the following references apply:

[1] ETSI TR 121 905: "Universal Mobile Telecommunications System (UMTS); Vocabulary for 3GPP Specifications (3GPP TR 21.905)".

[2] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

NOTE: Available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[3] ETSI TR 102 397-1: " Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs; Part 1: Common Mapping".

[4] ETSI ES 202 391-13: "Open Service Access (OSA); Parlay X Web Services; Part 13: Address List Management".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 102 397-1 [3] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 102 397-1 [3] apply.

4 Mapping description

The Terminal Status capability can be implemented with Parlay/OSA Mobility User Status.

It is applicable to ETSI OSA 1.x/2.x/3.x, Parlay/OSA 3.x/4.x/5.x and 3GPP Releases 4 to 6.

5 Sequence diagrams

5.1 Single address query

To query the terminal status for a single address, the synchronous request results in an asynchronous request being made to the User Status SCF to retrieve the status, and the result translated to return to the Enterprise Application.

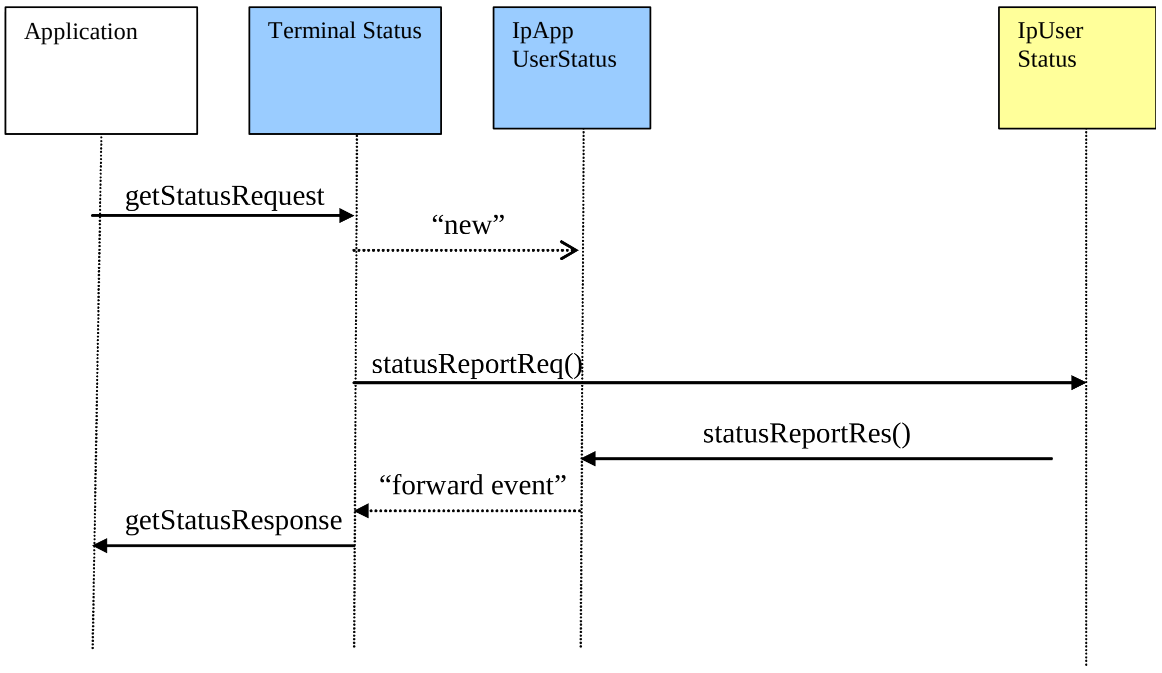

Figure 1

5.2 Group query

A query of the terminal status for a group of addresses (using either application or network managed groups) requires the service to create the set of terminals for which the request applies, then interacting with the User Status SCF to retrieve the information for the set of terminals.

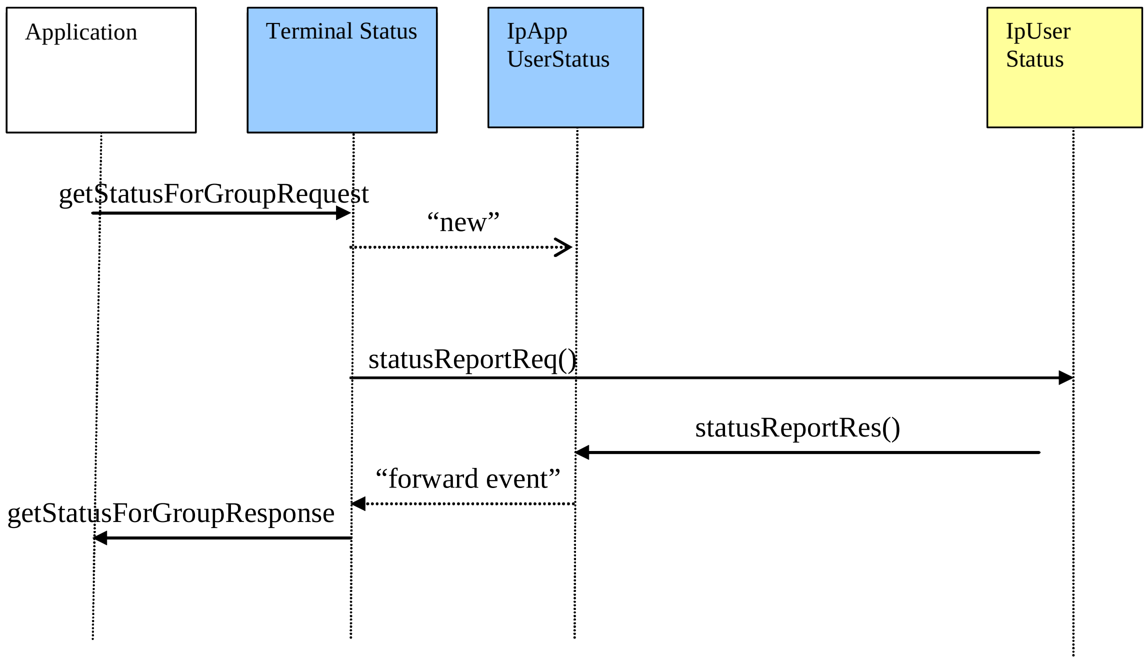

Figure 2

5.3 Notification

Notifications of change in terminal status may be made by setting up a notification with the User Status SCF and managing those notifications to provide the appropriate notifications and content to the Enterprise Application. In the following sequence diagram, the yellow highlighted sub-sequence represents optional actions initiated by the Terminal Status web service, if the **checkImmediate** flag in the **startNotificationRequest** message is enabled.

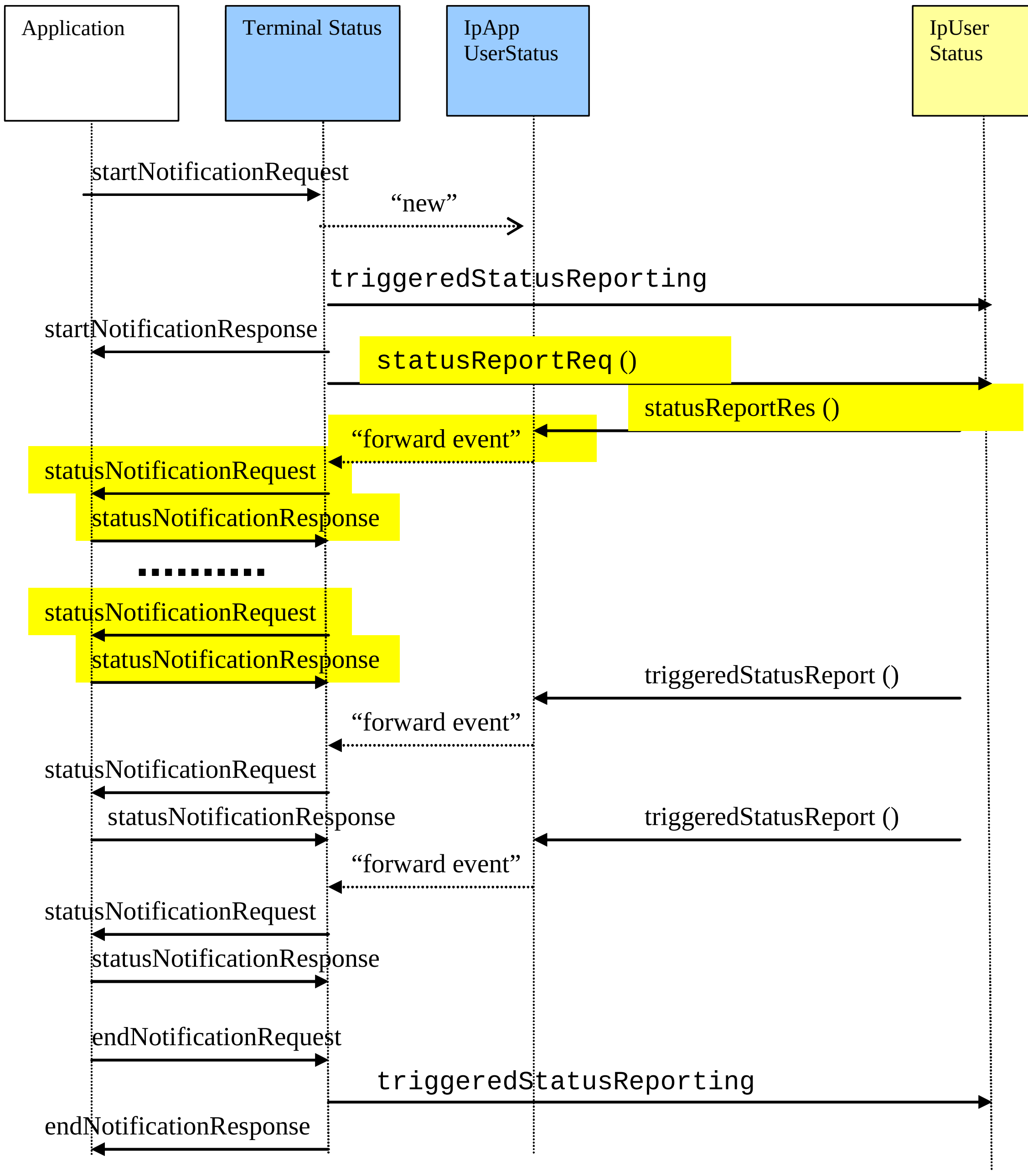

Figure 3

6 Detailed mapping information

6.1 Operations

6.1.1 getStatus

The sequence diagram in clause 5.1 illustrates the flow for the **getStatus** operation.

The **getStatus** operation is synchronous from the Parlay X client's point of view. It is mapped to the following Parlay/OSA methods:

- `IpUserStatus.statusReportReq;`
- `IpAppUserStatus.statusReportRes;`
- `IpAppUserStatus.statusReportErr.`

6.1.1.1 Mapping to `IpUserStatus.statusReportReq`

The `IpUserStatus.statusReportReq` operation is invoked with the following parameters.

Name	Type	Comment
appStatus	IpAppUserStatusRef	Reference to callback (internal).
users	TpAddressSet	Populated with one `TpAddress` element, constructed based on the URI provided in **address** part of **getStatusRequest**, mapped as described in in TR 102 397-1 [3].

The result from `IpUserStatus.statusReportReq` is used internally to correlate the callbacks.

Parlay exceptions thrown by `IpUserStatus.statusReportReq` are mapped to Parlay X exceptions as defined in clause 6.2.2.

6.1.1.2 Mapping from `IpAppUserStatus.statusReportRes`

When status information is available, the `IpAppUserStatus.statusReportRes` callback is invoked. It is expected to contain a `TpUserStatus` element whose `UserID` field matches the `TpAddress` passed to the status request operation. The fields of the `TpUserStatus` element are mapped to the **result** part of the **getStatusResponse** message as follows.

Name	Type	Comment
UserID	TpAddress	Matches the `TpAddress` element passed to the status request operation.
StatusCode	TpMobilityError	If the value is `P_M_OK`, a result is returned, otherwise an exception is thrown by **getStatus** using the mapping from `TpMobilityError` values to Parlay X Exceptions as defined in clause 6.2.1.
Status	TpUserStatusIndicator	If the `StatusCode` value (above) is `P_M_OK`, then the `Status` value is mapped to the **Status** enumeration as follows: `P_US_REACHABLE` -> **Reachable** `P_US_NOT_REACHABLE` -> **Unreachable** `P_US_BUSY` -> **Busy**.
TerminalType	TpTerminalType	Not mapped.

6.1.1.3 Mapping from `IpAppUserStatus.statusReportErr`

If an error prevents the status information from being reported, the `IpAppUserStatus.statusReportErr` callback is invoked. If this occurs, an exception will be thrown by **getStatus** based on the value of the `cause` parameter. Refer to the `TpMobilityError` to Parlay X exception mapping as defined in clause 6.2.1.

6.1.2 getStatusForGroup

The sequence diagram in clause 5.2 illustrates the flow for the **getStatusForGroup** operation.

The **getStatusForGroup** operation is synchronous from the Parlay X client's point of view. It is mapped to the following Parlay/OSA methods:

- `IpUserStatus.statusReportReq;`
- `IpAppUserStatus.statusReportRes;`
- `IpAppUserStatus.statusReportErr.`

6.1.2.1 Mapping to `IpUserStatus.statusReportReq`

The `IpUserStatus.statusReportReq` operation is invoked with the following parameters.

Name	Type	Comment
appStatus	IpAppUserStatusRef	Reference to callback (internal).
users	TpAddressSet	Populated with `TpAddress` elements, constructed based on: • The individual address URIs passed in the **addresses** part of **getStatusForGroupRequest**. • The address URIs obtained by resolving group URIs in the **addresses** part into individual address URIs. URI to `TpAddress` mapping is described in TR 102 397-1 [3].

The result from `IpUserStatus.statusReportReq` is used internally to correlate the callbacks.

Parlay exceptions thrown by `IpUserStatus.statusReportReq` are mapped to Parlay X exceptions as defined in clause 6.2.2.

6.1.2.2 Mapping from `IpAppUserStatus.statusReportRes`

When status information is available, the `IpAppUserStatus.statusReportRes` callback is invoked. It is expected to contain a `TpUserStatus` element for each requested TpAddress in the original request. The `TpUserStatus` elements are used to create an array of **StatusData** elements comprising the **result** part of the **getStatusForGroupResponse** message. The mapping between `TpUserStatus` and **StatusData** is as follows.

TpUserStatus element		StatusData element		
Name	**Type**	**Name**	**Type**	**Comment**
UserID	TpAddress	**address**	**anyURI**	`TpAddress` to URI mapping is described in TR 102 397-1 [3].
StatusCode	TpMobilityError	**reportStatus**	**Retrieval Status**	`StatusCode` value is mapped to **reportStatus** as follows: `P_M_OK` -> **Retrieved** `P-M-xxx` (i.e. all other values) -> **Error**.
Status	TpUserStatus Indicator	**currentStatus**	**Status**	If the `StatusCode` value (above) is `P_M_OK`, then the mapping is as follows: `P_US_REACHABLE` -> **Reachable** `P_US_NOT_REACHABLE` -> **Unreachable** `P_US_BUSY` -> **Busy**.
TerminalType	TpTerminalType			Not mapped.
		error Information	**common: ServiceError**	If the `StatusCode` value (above) is **NOT** `P_M_OK`, then the value of this element is defined in clause 6.2.1.

In the event that a a `TpUserStatus` element is missing for a requested **address** in the original request, then a **StatusData** element (with a **reportStatus** value = **NotRetrieved**) is included in the **result** part of the **getStatusForGroupResponse** message.

6.1.2.3 Mapping from `IpAppUserStatus.statusReportErr`

If an error prevents the status information from being reported, the `IpAppUserStatus.statusReportErr` callback is invoked. If this occurs, an exception will be thrown by **getStatusForGroup** based on the value of the `cause` parameter. Refer to the `TpMobilityError` to Parlay X exception mapping as defined in clause 6.2.1.

6.1.3 startNotification

The sequence diagram in clause 5.3 illustrates the flow of events when a client establishes a triggered status notification request.

The **startNotification** operation is mapped to the following Parlay/OSA methods:

- `IpUserStatus.triggeredStatusReportingStartReq;`
- `IpUserStatus.statusReportReq.`

6.1.3.1 Mapping to `IpUserStatus.triggeredStatusReportingStartReq`

The **Endpoint** reference provided by the client in the **reference** part of the **startNotificationRequest** message will be used by the Parlay X implementation to locate the client's **statusNotification**, **statusError** and **statusEnd** operations when network events occur that trigger notifications.

The string passed by the client in the **Correlator** element of the **reference** part will be passed back to the client when the client's **statusNotification**, **statusError** and **statusEnd** operations are invoked as a result of status changes in the network that match this notification request.

The `IpUserStatus.triggeredStatusReportingStartReq` operation is invoked with the following parameters.

Name	Type	Comment
appStatus	IpAppUserStatusRef	Reference to callback (internal)
users	TpAddressSet	Populated with `TpAddress` elements. These are constructed based on: • The individual address URIs passed in the **addresses** part. • The address URIs obtained by resolving group URIs in the **addresses** part into individual address URIs. URI to `TpAddress` mapping is described in TR 102 397-1 [3].

The result from `IpUserStatus.triggeredStatusReportingStartReq` is stored internally and associated with the string passed by the client in the **Correlator** element of the **reference** part.

Exceptions thrown by `IpUserStatus.triggeredStatusReportingStartReq` are mapped to Parlay X exceptions as defined in clause 6.2.2.

6.1.3.2 Mapping to `IpUserStatus.statusReportReq`

If the **checkImmediate** part of the **startNotificationRequest** message is set to a value of "True", then the `IpUserStatus.statusReportReq` operation is invoked with the following parameters.

Name	Type	Comment
appStatus	IpAppUserStatusRef	Reference to callback (internal)
users	TpAddressSet	Populated with `TpAddress` elements, constructed based on: • The individual address URIs passed in the **addresses** part. • The address URIs obtained by resolving group URIs in the **addresses** part into individual address URIs. URI to `TpAddress` mapping is described in TR 102 397-1 [3].

The result from `IpUserStatus.statusReportReq` is used internally to correlate the callback: i.e. `IpAppUserStatus.statusReportRes/Err`.

Parlay exceptions thrown by `IpUserStatus.statusReportReq` are ignored and not reported to the application over the Parlay X interface. Similarly, if the `IpAppUserStatus.statusReportErr` method callback is invoked, the Parlay X application is not notified.

Note that, although the secondary "checkImmediate" feature of the **startNotification** operation has failed, the primary feature may be operational.

6.1.4 endNotification

The sequence diagram in clause 5.3 illustrates the flow of events when a client establishes a triggered status notification request (and subsequently ends it).

The **endNotification** operation is mapped to the following Parlay/OSA methods:

- `IpUserStatus.triggeredStatusReportingStop.`

6.1.4.1 Mapping to `IpUserStatus.triggeredStatusReportingStop`

The `IpUserStatus.triggeredStatusReportingStop` operation is invoked with the following parameters.

Name	Type	Comment
stopRequest	TpMobilityStopAssignmentData	
The fields of this structure are populated as follows:		
stopRequest.AssignmentId	TpSessionID	Set to the `TpSessionID` value associated with the string passed by the client in the **correlator** part of **endNotificationRequest**.
stopRequest.StopScope	TpMobilityStopScope	Set to `P_M_ALL_IN_ASSIGNMENT.`
stopRequest.Users	TpAddressSet	Not populated.

Exceptions thrown by `IpUserStatus.triggeredStatusReportingStop` are mapped to Parlay X exceptions as defined in clause 6.2.2.

6.1.5 statusNotification

The sequence diagram in clause 5.3 illustrates the flow of events when a client establishes a triggered status notification request (and an event subsequently occurs to trigger a status notification).

The **statusNotification** operation is mapped from the following Parlay/OSA methods:

- `IpAppUserStatus.triggeredStatusReport;`
- `IpAppUserStatus.statusReportRes.`

6.1.5.1 Mapping from `IpAppUserStatus.triggeredStatusReport`

The **statusNotification** operation is invoked by the Parlay X implementation when a status change occurs in the network that matches a client's request for triggered status. It is invoked in direct response to a Parlay `IpAppUserStatus.triggeredStatusReport` invocation. The parameter mapping is as follows.

IpAppUserStatus.triggeredStatus Report		**statusNotification**		
Name	**Type**	**Name**	**Type**	**Comment**
assignmentID	TpSessionID	**correlator**	**string**	Parlay X implementation may use the assignmentID to determine the **correlator** value to return.
status	TpUserStatus	Elements are mapped as follows:		
status.UserID	TpAddress	**address**	**anyURI**	**address** URI constructed as described in TR 102 397-1 [3].
status.Status Code	TpMobilityError			Not mapped. If `TpMobilityError` is any value other than `P_M_OK`, then the **statusError** operation is invoked instead, as described in clause 6.1.6.
status.Status	TpUserStatus Indicator	**current Status**	**Status**	The mapping is as follows: `P_US_REACHABLE` -> **Reachable** `P_US_NOT_REACHABLE` -> **Unreachable** `P_US_BUSY` -> **Busy**.
status.Terminal Type	TpTerminalType			Not mapped.

6.1.5.2 Mapping from `IpAppUserStatus.statusReportRes`

If the **checkImmediate** part of a **startNotificationRequest** message was set to a value of "True" (reference 6.1.3.2) then, when status information is available, the `IpAppUserStatus.statusReportRes` callback is invoked. It is expected to contain a `TpUserStatus` element for each requested TpAddress. Each `TpUserStatus` element is mapped to a separate **statusNotificationRequest** message. The parameter mapping is as follows.

IpAppUserStatus. StatusReportRes		**statusNotificationRequest** message part		
Name	**Type**	**Name**	**Type**	**Comment**
assignmentID	TpSessionID	**correlator**	**string**	Parlay X implementation may use the assignmentID to determine the **correlator** value to return.
status	TpUserStatus	Elements of `status` are mapped as follows:		
status.UserID	TpAddress	**address**	**anyURI**	**address** URI constructed as described in TR 102 397-1 [3].
status.Status Code	TpMobilityError			Not mapped. If `TpMobilityError` is any value other than `P_M_OK`, then the `TpUserStatus` element is ignored: no **statusNotificationRequest** message is sent for this **address**.
status.Status	TpUserStatus Indicator	**current Status**	**Status**	The mapping is as follows: `P_US_REACHABLE` -> **Reachable** `P_US_NOT_REACHABLE` -> **Unreachable** `P_US_BUSY` -> **Busy**.
status.Terminal Type	TpTerminal Type			Not mapped.

6.1.6 statusError

The sequence diagram in clause 5.3 illustrates the flow of events associated with the triggered status notification capability.

The **statusError** operation is mapped from the following Parlay/OSA methods:

- `IpAppUserStatus.triggeredStatusReportErr.`

6.1.6.1 Mapping from `IpAppUserStatus.triggeredStatusReportErr`

The **statusError** operation is invoked by the Parlay X implementation when an error condition occurs that results in the termination of the entire notification request by the Parlay X implementation. It is invoked in response to a Parlay `IpAppUserStatus.triggeredStatusReportErr` invocation. The parameter mapping is as follows.

IpAppStatus.triggeredStatus ReportErr		**statusError**		
Name	**Type**	**Name**	**Type**	**Comment**
assignmentID	TpSessionID	**correlator**	**string**	Parlay X implementation may use the assignmentID to determine the **correlator** value to return.
cause	TpMobilityError	**reason**	**string**	**reason** string is constructed based on the value of the cause and/or diagnostic parameter(s).
diagnostic	TpMobility Diagnostic			

6.1.7 statusEnd

The **statusEnd** notification is called when the notification ends due to the end of the duration being met, or when the count of notifications has been delivered. The **statusEnd** notification does not occur when the notification is deliberately ended or in the case of an error. There is no mapping from Parlay/OSA for this capability.

6.2 Exceptions

6.2.1 Mapping from `TpMobilityError`

The following table indicates how `TpMobilityError` values are mapped to Parlay X exceptions.

Value	**Service Exception**	**Notes**
P_M_SYSTEM_FAILURE	SVC0001	With error number
P_M_UNAUTHORIZED_NETWORK	SVC0001	With error number
P_M_UNAUTHORIZED_APPLICATION	SVC0001	With error number
P_M_UNKNOWN_SUBSCRIBER	SVC0002	
P_M_ABSENT_SUBSCRIBER	SVC0002	
P_M_POSITION_METHOD_FAILURE	SVC0001	With error number

6.2.2 Mapping from Parlay/OSA method exceptions

For the present document, the mapping of Parlay/OSA API method exceptions to Parlay X Web Service exceptions is common and defined in TR 102 397-1 [3]. There are no service-specific exception mappings.

7 Additional notes

No additional notes.

History

Document history		
V1.1.1	August 2005	Publication