Telco's Application Store

Approved Version 1.0 – 11 Aug 2015

Open Mobile Alliance

OMA-ER-TAS-V1_0-20150811-A

Contents

Figures

Tables

1. Scope

This Enabler Release (ER) document is a combined document of requirements, architecture and technical specification for Telco's Application Store Enabler. The TAS Enabler is expected to provide functions of APIs Management, Developer Management, Application Management and Application Creation Support.

2. References

2.1 Normative References

[OSE] "OMA Service Environment", Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC4234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. October 2005, URL:http://www.ietf.org/rfc/rfc4234.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

2.2 Informative References

[ISO-3166 Alpha-2] URL:http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

[OMADICT] "Dictionary for OMA Specifications", Version 2.7, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_7, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Application	(see [OMADICT]) An application may consist of only a software on the device, or it may be associated to a server side software (managed by a service provider) to provide a service to the end user.
Application Category	Application Category is a mechanism to divide Applications into different groups.
Developers Community	Developers Community is a platform, which provides resources and tools to developers for developing Applications. It mainly consists of functionalities to manage Applications, developers and developing resources and tools. After registered to Developers Community, developers could submit Applications to it.
In Application Purchase	In Application Purchase is a function which allows a developer to embed items within an application.

3.3 Abbreviations

CRAPI	Capability Resource API
IAP	In Application Purchase
OMA	Open Mobile Alliance

4. Introduction

Application stores offer downloadable applications to an audience of mainly consumers, via a store front, which is either embedded in the mobile handset or found on the Web (for both mobile as well as fixed devices).

Existing commercial appstores have attracted over 100,000 different applications and billions of downloads.

Handset and operating system (OS) manufacturers looking to create similar excitement with their phones and/or OSs have also introduced application stores (Source: Gartner July 2009).

In this scenario, the Telcos do not have an important role in the content or application sales transactions, mainly providing the role of network access or broadband provider. Although Telcos benefit from an increase in mobile data usage, they lack the control over purchasing through application stores.

By opening their own network resource and investing in an open platform for developers, Telcos will be able to offer the applications to enrich the user experience and create stickiness and drive long term customer loyalty.

Some Telcos have come to recognize they may be better served by opening their network intelligence and back-office software platforms to the application development community. Thus, they no longer must depend on one or two killer applications to drive revenue, but instead deliver a compelling application platform, enabling thousands of high quality applications to be produced, while at the same time, dramatically improving customer satisfaction. But there is no standard for this solution.

The TAS Enabler aims to establish a unified framework of Application Store which integrates all the stages of application development support, application distribution, application sales and so on. And by TAS, Telcos can be in the leadership of the application store ecosystem.

TAS can create an application platform that would be attractive to the developer and content partner community:

- Capability discovery and binding
- Purchase Convenience
- Distribution and Reach
- Multi-device platform Support
- Operational Support (e.g. application upgrade)

This document includes requirement, architecture, and technical specifications.

4.1 Version 1.0

The TAS Enabler addresses the following four work areas:

- Developer Management. This area mainly addressed functions to enable application developers to register to TAS, and functions for TAS to manage those developers.
- APIs management
    - APIs life cycle management, APIs publishing (no new APIs will be defined)
- Application management
    - Application state management (Re-using OSPE. OSPE is re-used as a start point, and TAS could add specific life cycle status)
    - Application display and sales function
    - Application distribution/installation management
- Application Creation Support, including development and test support for applications

The Security and Charging aspects will also be considered during the specification development.

5. Requirements (Normative)

The following sections describe the requirements for: Application Management, Developer Management, Application Creation Support, In-Application Purchase, Charging, Privacy, and Security.

5.1 Application Management

5.1.1 Application Information

The TAS Enabler SHALL be able to manage different types of Applications, including but not limited to:

- Games
- Contents (e.g. themes for mobile phones)
- Tools (e.g. web browser)
- Entertainment
- Books

The TAS Enabler SHALL be able to store the Application Information, including but not limited to:

- Application ID
- Name
- Developer
- Type
- Description
- Status
- Version
- Submit time
- Effective Date (the date on which an Application begins to be available to Users)
- Expire Date
- Price

5.1.2 Application Uploading and Auditing

The TAS Enabler SHALL allow developers to upload Applications and related information, such as Application name, type, keywords, proposed price, publish date, expiration date and charging option, etc.

The TAS Enabler SHALL be able to generate an application ID for the uploaded / audited application.

The TAS Enabler SHALL be able to audit uploaded Applications and audit related information (e.g. name of the developer) according to the TAS service provider's policies. The auditing process on uploaded Applications includes security checking, such as virus scanning and content screening. The auditing process on related information is to check whether the information submitted by the developer is authentic and complete.

The TAS enabler SHOULD notify the developers of the outcome of the audit.

The TAS Enabler SHALL enable the TAS service provider to configure the policies which are used to audit Applications and related information.

The TAS Enabler SHALL support developers to submit multiple variants of the same application. (e.g. an application might have a version for vendor A's smartphone and another version for vendor A's feature phone)

The TAS Enabler SHALL support developers to update their own uploaded applications (e.g. submit a new version, including the software and related informations) in order to improve the functions or add some new features. The new version may replace an old version, or they may both exist.

The TAS Enabler SHALL support developers to manage (e.g. view application's status, modify related information, and delete their applications, etc) their own uploaded applications. A developer can search and view metadata associated to other developers' applications.

5.1.3 Application Test

After uploaded and successfully audited, the application may be tested to check whether it has a bright market future for formal business. The Application Test is only open for beta-testers in the Developer Community. Beta-testers download and use the application for a period of time. Then beta-testers fill out the questionnaire to provide feedback on their experience using the application. The TAS service providers may evaluate the result of questionnaire and if the result shows that beta-testers like to use this application, the TAS service provider may decide for the application to access formal business processes.

The TAS Enabler MAY enable developers to request internal testing of their submitted applications in Developers Community.

The TAS Enabler MAY enable developers to design questionnaire for the internal testing.

The TAS Enabler MAY support the TAS service provider to recruit beta-testers to experience the internal testing application.

The TAS Enabler MAY enable beta-testers to download the internal testing application and related questionnaire.

The TAS Enabler MAY enable beta-testers to submit feedback using the questionnaires.

The TAS Enabler MAY enable developers to view the questionnaires result of their own applications which are under internal testing.

The TAS Enabler MAY support the TAS service provider to decide whether to publish the internal tested applications according to the questionnaires result.

The TAS Enabler MAY notify the developers of the outcome of the tests.

5.1.4 Application Distribution

The TAS Enabler SHALL allow Applications to be downloaded.

The TAS Enabler SHALL enable the user to know the status (e.g. successful download, abandoned download) of Application downloading.

The TAS Enabler SHALL support dynamically generating information related to the Applications (e.g. download location, usage rights).

The TAS Enabler SHALL support logging of Application downloads.

The TAS Enabler SHALL support reliable underlying transport delivery mechanisms and SHOULD support resuming failed Application downloads.

The TAS Enabler SHALL control whether and how Applications can be distributed after they are downloaded to Users' terminals.

5.1.5 Application Installation

The TAS Enabler SHALL be notified about the status (e.g. installed, uninstalled) of Application installation.

5.1.6 Application Recommendation

The TAS Enabler SHALL be able to provide a list of recommended Applications (e.g. application popularity, overall usage history, developer's popularity).

The TAS Enabler SHALL be able to provide a list of Applications associated to a specific Application (e.g. based on categories, key words).

5.1.7 Application Feedback

The TAS Enabler SHOULD allow Users to provide feedback (e.g. rate and comments) on Applications they have used. For example, the rating on an application could be a score from 1 to 5.

5.1.8 Display application

The TAS Enabler SHALL be able to access device capabilities information.

When available, the device capabilities (e.g. screen size, screen resolution, operating system) information SHALL be used by TAS Enabler when providing the Application lists.

5.1.9 Application Category Management

The TAS Enabler SHOULD support TAS service providers to define a list of possible Application Categories. The Application Category includes (not exhaustive): category name, category id, description element(s), etc.

The TAS Enabler SHOULD allow a developer to assign an Application Category (among the Application Categories made available by the TAS service provider) to his applications.

The TAS Enabler SHOULD allow the Service Provider to identify if an Application Category (as assigned by the developer) is not appropriate for a specific application.

5.1.10 Blacklists for Applications and developers

The TAS Enabler SHALL enable users to report malicious applications to the TAS service providers.

The TAS Enabler SHALL be able to allow TAS service providers to verify and then add / delete malicious applications or their developers to / from Blacklists according to TAS service provider's policy (e.g. user's evaluations or complaints).

Once the malicious applications or their developers are added to the Blacklist, those applications cannot be displayed to or accessed by users.

5.1.11 Application Search

The TAS Enabler SHOULD be able to allow users, TAS service providers and developers to search Applications by related attributes: e.g. application name, application category, keywords, published date, expiration date, etc.

5.1.12 Application revocation

The TAS Enabler SHALL be able to revoke from the TAS Server the applications that are already uploaded based on TAS service provider's policy.

5.1.13 Sorting Mechanism

The TAS Enabler SHOULD be able to provide the sorting mechanism for applications so that a user can decide whether or not to use it and which criteria to apply on, such as, the latest version, the highest download rates within a specific time period, the price, etc.

5.1.14 Update Notification

The TAS Enabler SHALL support TAS service providers to provide update notification to a user when there is an update of the available applications. The TAS Enabler SHALL allow users to subscribe/unsubscribe to such update notifications.

The update notification might be an SMS, an email, a pop-up message, a voice mail, etc.

5.1.15 Application State Management

The TAS Enabler SHALL be able to manage different states of Applications, including but not limited to:

- Submitted
- Audited
- Tested
- Online
- Offline
- End(Application is deleted from TAS)

The TAS Enabler SHALL support application state flow control.

The TAS Enabler SHALL guarantee that only those Applications of online state may be downloaded.

The TAS Enabler SHALL support a mechanism for the Developer to be notified about the status of its Application.

The notification of the application status can be triggered either by state transition or by developer's proactive request.

5.1.16 In-Application Purchase Management

The TAS Enabler MAY provide the In Application Purchase functionality.

The IAP functionality allows developers to embed a store in their applications. The IAP functionality enables developers to manage the payment information (e.g. price, payment period, etc.) of their IAP items.

An IAP item is an item that a developer wants to sell in an application's store. It's associated with the application. There are several supported kinds of IAP items that may be sold using In App Purchase functionality, for example:

1. Content: including digital books, magazines, photos, artwork, game levels, game characters, and other digital content that can be delivered within applications.
2. Features: products unlock or advanced features which have already been delivered as added part of the applications (e.g. a game with multiple smaller games that could be purchased by the user).

The developer could use In App Purchase to implement the following scenarios (not limited to these):

1. A basic version of application with additional premium features.
2. A book reader application that allows the user to purchase and download new books.
3. A game that offers new environments (levels) to explore.

4. An online game that allows the player to purchase virtual property.

5.2 Developer Management

5.2.1 Categories of developers

The TAS Enabler SHALL allow the TAS service provider to apply different policies on developers based on their category.

Developers of the TAS Enablers can be assigned to different categories, for example they can be divided into two categories: individual developers and enterprise developers.

Individual developers are registered to the TAS Enabler, permitted to access resources in the Developers Community for use of application developing. An individual developer could be affiliated to one enterprise developer.

Enterprise developers are registered to the TAS Enabler, permitted to access resources in the Developers Community for use of application developing. Enterprise developers provide Applications to the TAS Enabler.

5.2.2 Developer's information

For individual developers, their information includes (not exhaustive): category, name, password, personal ID number, phone number, email address, bank account information, level and affiliated Enterprise.

For enterprise developers, their information includes (not exhaustive): category, name, password, address, license number, amount of registered capital, bank of registered capital, bank account, web site address, contact name, contact email address, contact phone number and level.

Both individual and enterprise developers have the capability to manage their information.

5.2.3 Developer State Management

The TAS Enabler SHALL enable both individuals and enterprises to register to the TAS Enabler.

After registration, they are called developers.

The TAS Enabler SHALL enable developers to de-register from the TAS Enabler.

The TAS Enabler SHOULD be able to manage different states of developers.

5.2.4 Developer Contract

While registered developers have access to resources in the Developers Community, they should contract with TAS service provider before they could submit Applications to it. Developers submit a contract request to the TAS service provider in order to initiate collaboration with the TAS service provider. If the contract request is approved by the TAS service provider, the contract between the developer and the TAS service provider is valid. A valid contract consists of information of the developer, the model of cooperation and expiration date, etc. The contracted developer may request to revoke a contract before its expiration date by submitting a contract revocation to the TAS service provider.

5.2.5 Settlement

The TAS Enabler SHALL support settlement with developers based on pre-defined settlement policies.

Settlement policies include:

- when to start a settlement process (e.g. specific time interval, threshold of amount);
- ratio of revenue sharing (e.g. on a 30%/70% basis);
- etc.

The TAS Enabler SHALL be able to generate settlement report to developers according to related contract and transaction records.

The TAS Enabler SHOULD enable developers to check the sales of their applications. The list of sales MAY be sorted (e.g. based on a specific period of time and application type).

5.2.6 Blacklists for Developers

Once developers are in the Blacklist, TAS Enabler MAY be able to apply some punishment mechanisms (such as penalty, registration denial permanently or within certain period, etc.)

5.3 User Management

5.3.1 User Information

The TAS Enabler SHALL be able to manage different information of users, including but not limited to:

- Essential information (e.g. identifier, name)
- Paid account (e.g. MSISDN, bank card)
- Payment type (e.g. post-paid , prepaid)
- TAS Status (e.g. active, inactive)

5.3.2 User Purchasing Management

The TAS Enabler SHALL allow users to purchase applications and In Application Purchase Items.

The TAS Enabler MAY allow users , service providers and developers to inquire user's purchase history according to transaction information (e.g. transaction id, identifier, name, Application id, time, transaction status), according to Service Provider and user's policy.

5.3.3 Application Favorites

Application favourites can help a user keep his favorite applications and so on.

The TAS Enabler MAY allow a user to manage his favorite applications.

5.3.4 User Recommendation

The TAS Enabler SHALL allow users to recommend applications to other users.

The TAS Enabler SHALL support users to receive other users' recommendation and evaluation of applications when users browse the applications.

5.3.5 Personalized Display

The TAS Enabler SHALL be able to display to a User a list of Applications based on specific criteria (e.g. purchase history, application favourites, type, key words).

5.3.6 Application Shopping Cart

Application shopping cart can help a user purchase multiple applications those he favourites together with one payment.

The TAS Enabler SHALL allow a user to add the applications those he wants to purchase to the application shopping cart.

The TAS Enabler SHALL allow a user to delete the applications from the application shopping cart.

The TAS Enabler SHALL allow a user to add his favorite applications from the application favorites to the shopping cart.

The TAS Enabler SHALL allow a user to select the final applications from the shopping cart with one payment.

5.4 Application Creation Support

5.4.1 Application Creation Guidance

The TAS Enabler SHALL give a guidance to the developers about the steps to create an application in TAS (e.g. apply for an applicationID before development).

The TAS Enabler SHALL allow the developers to know what kind of applications (i.e. offline or online applications, applications fit for which smartphone operating system) can be accepted by TAS, and according to the specific kind of application what services and resources can they obtain from TAS.

5.4.2 Capability Resource Support

The TAS Enabler SHALL allow the developers to discover what capability resources are available for them and information to use those resources (e.g. WSDL documents for WebService APIs).

The TAS Enabler SHALL allow the developers to get authorization from capability resource providers or service providers (depending on the business model) to use their resources.

5.4.3 Application Development Support

The TAS Enabler MAY provide API SDKs (fit for specific development language) of registered capability resource and corresponding documentation

The TAS Enabler MAY provide online development environment. Registered developers can login to the online development environment and easily develop an application online.

5.4.4 Application Test Support

The TAS Enabler MAY provide simulation test tools that can be used to illustrate the validity of API invoking and to demonstrate terminal display.

The TAS Enabler MAY provide online test environment that enable developers to test their applications with a pre-approved test account.

5.5 Capability Resources (APIs) Management

The capability resources include operator's network resources (e.g. sending an SMS/MMS message) and internet resources (e.g. facebook's user information exposed by its open REST interfaces). Those resources are usually presented as APIs. They could be registered to the TAS Enabler, and be discovered by external entities. Developers can use capability resources by invoking the APIs.

The TAS Enabler SHALL allow capability resources providers to register their capability resources with their information to the TAS Enabler.

The TAS Enabler SHALL allow capability resources providers to update the information of their registered capability resources.

The TAS Enabler SHALL allow capability resources provider to de-register their registered capability resources.

The TAS Enabler SHALL be able to store the information of registered capability resources.

The TAS Enabler SHALL be able to manage the life cycle status of registered capability resources.

The TAS Enabler SHALL allow external entities (e.g. developers) to query the information of registered capability resources.

The TAS Enabler SHALL allow the TAS Service Provider to manage (i.e. to register, audit, publish and delete) capability resources.

The TAS Enabler SHALL allow the capability resource providers or service providers (depending on the business model) to set the authorization mechanisms for their resources.

5.6 Charging

The TAS Enabler SHALL support for Online Charging and Offline Charging.

The TAS Enabler SHALL support the creation of Charging Events needed for different charging scenarios, e.g. charging only in case of successful downloading of an application, no charging for experimenting with new application, charging based on service subscriptions.

The TAS Enabler SHALL support the creation of Charging Events to charge users for in-application purchase.

The TAS Enabler SHALL support the creation of Charging Events to charge developers for capability resource purchase.

The TAS Enabler SHALL be able to collect application charging records.

TAS Enabler SHALL be able to support charging according to the user's identifier. User can choose the way of payment in accordance with its identifier.

The TAS Enabler SHALL allow a user A to buy an application for another user B, subject to policy control by the TAS service provider (e.g. user B may not want this, user A may have an upper limit, the TAS service provider may not allow this); the TAS Enabler SHALL support notification to User B that User A has bought an application for him/her.

Note: an application might offer monthly access to financial information or to an online game portal. In this case, the developer will be responsible for both tracking subscription expirations and renewal billing for contents/application usage; the TAS Enabler does not monitor subscription duration and does not offer an automatic billing mechanism.

5.7 Privacy

The TAS Enabler SHOULD notify users while downloading applications that may invoke some operations relative to user's privacy (e.g. notifying users that the applications may invoke user's address book, access the network or make a call, etc. Notification information may be displayed while browsing or downloading the applications).

The TAS Enabler SHOULD be able to block the download of applications based on user's privacy preferences.

The TAS Enabler SHOULD allow to set and change user's privacy preferences.

5.8 Security

5.8.1 Authentication

The TAS Enabler SHALL be able to identify and authenticate the developers for the management (e.g. updating/uploading of applications) of the applications.

The TAS Enabler SHOULD be able to identify and authenticate each user before the users may download an application.

The identifier used for the authentication SHALL be unique for a specific TAS service provider such as an EMAIL address, or MSISDN.

5.8.2 Authorization

The TAS Enabler SHALL provide protection for applications downloaded from the TAS store to prevent unauthorized distribution.

The TAS Enabler SHALL be able to authorize the developers to manage (e.g. update/upload) the applications.

5.9 Statistics

The TAS Enabler MAY be able to support statistics function, such as the sum of developers in TAS, the sum of applications in TAS, the gross revenue of one specific application, etc.

6. Architectural Model

6.1 Dependencies

No dependencies are identified in this release.

6.2 Architectural Diagram

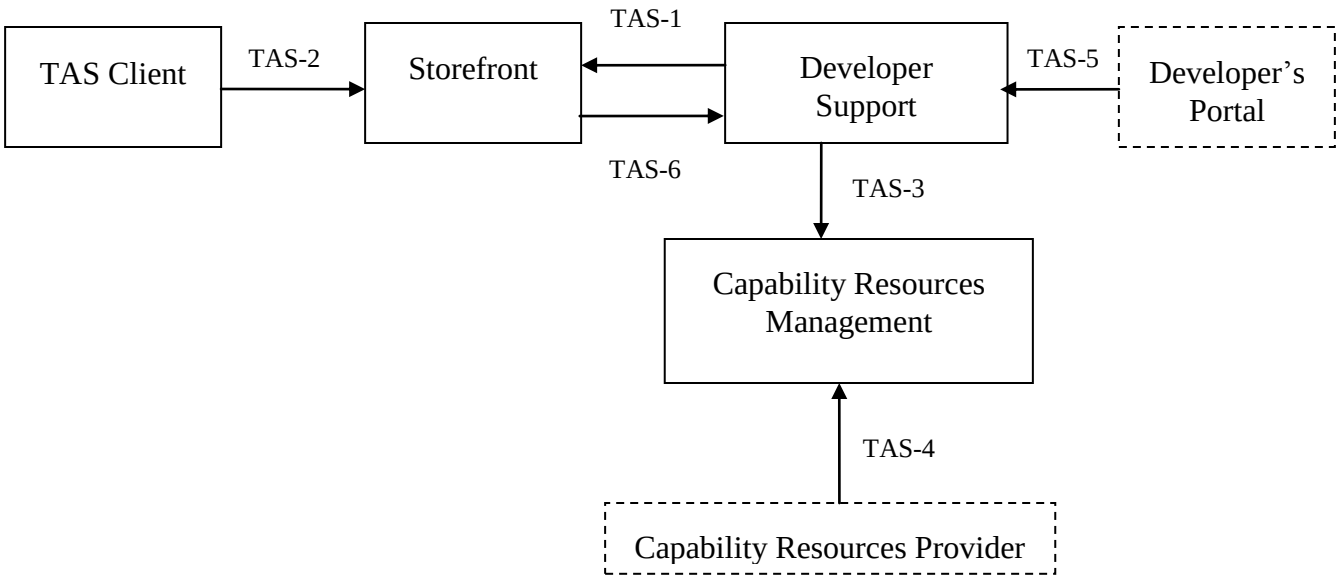

Figure 1: Architectural Diagram

6.3 Functional Components and Interfaces/reference points definition

6.3.1 Functional Components

6.3.1.1 Internal Functional Components

6.3.1.1.1 Storefront

The Storefront component is responsible for providing Applications to users. Users download the Applications by using the TAS Client, which is embedded in their handsets or installed in PC, or by accessing a web portal. The Storefront component is responsible for User Management (such as user's personalisation data, user's purchase records, user's application favorites and shopping carts, etc.). Users can manage their personal information by the TAS Client or web portal.

The Storefront component is also responsible for providing IAP items to Application. It accepts the IAP purchase request from Application directly or indirectly (passed via TAS Client), then charge the User (need User confirmation) according to the information of the requested IAP items. Upon successfully charged, the Storefront responds the result to the Application.

The TAS Client interacts with the Store Front component via the TAS-2 interface.

The Storefront component is responsible for Application Management (such as Application Category Management, Application Sorting, Application Recommendation, etc.). The Applications should pass the audit process in the Developer Support component before they are submitted to the Storefront component via the TAS-1 interface.

TAS Storefront may provide user with applications that are compatible with the user's device capability.

6.3.1.1.2 Developer Support

The Developer Support component is responsible for developer management (such as developer state management, contract process, settlement process, etc) and management of Applications which are uploaded by developers via the TAS-5 interface. It audits the uploaded Applications and their related information and then makes them available to the Storefront component via the TAS-1 interface. The Applications may include features which need to invoke register resources. In this case, the information of the resources (e.g. how to invoke the resources) can be obtained via the TAS-4 interface.

The Developer Support component is also responsible for management of IAP item of uploaded Applications. It accepts IAP items configured to Applications and their information.

6.3.1.1.3 Capability Resources Management

The Capability Resources Management component is responsible for managing information of capability resources. The capability resources include operator's network resources and internet resources. The resources could be registered to the Capability Resources Management component with its information. The Capability Resources Management component also provides information of registered resources to other entities via the TAS-4 interface.

6.3.1.1.4 TAS Client

The TAS Client component is responsible for browsing and downloading Applications from the Storefront component, and interacting with the Storefront component to maintain the installation status of downloaded Applications.

TAS Client delivers the device capability information to the Storefront (e.g. when it requests to browse applications), by using the existed transport protocols, e.g.: HTTP User Agent Profile.

TAS Client is responsible for managing the Applications which are already downloaded and/or installed.

6.3.1.2 External Functional Components

6.3.1.2.1 Developer's Portal

The Developer's Portal component acts as a portal for Developers. Developers upload Application to the TAS Enabler using this component. This component interacts with the Developer Support component by using the TAS-5 interface, to manage Applications and IAP items, as well as to check the audit status of them.

6.3.1.2.2 Capability Resources Provider

The Capability Resources Provider component provides capability resources to the TAS Enabler. It registers capability resources and updates the information of those capability resources by the TAS-4 interface.

6.3.2 Interfaces

6.3.2.1 TAS-1

This interface is exposed by the Storefront component and can be used to accept audited Applications.

TAS-1 can be used to change the app's information in the Store Front according to the operation in Develop Support. For example, if the developer deletes one of his app which is online, the delete operation should be informed to the Store Front through TAS-1.

6.3.2.2 TAS-2

This interface is exposed by the Storefront component and can be used to accept download request of Applications.

TAS-2 can be used to report the download/install/uninstall operation of an application.

TAS-2 can be used to request purchase and return purchase result to user.

TAS-2 can be used to activate Users.

TAS-2 can be used to manage the User information.

TAS-2 can be used to log-in/log-out in Storefront.

TAS-2 can be used to request information of Applications.

TAS-2 can be used to purchase or gift, and to request application purchase records.

TAS-2 can be used to submit feedback of an application that user has downloaded.

TAS-2 can be used to request refund and return refund result to user.

6.3.2.3 TAS-3

This interface is exposed by the Capability Resources Management component and can be used to obtain information of registered capability resources.

TAS-3 can be used to inform the Developer Support about the operations of capability resource,

6.3.2.4 TAS-4

This interface is exposed by the Capability Resources Management component and can be used to register capability resources and update their information.

6.3.2.5 TAS-5

This interface is exposed by the Developer Support component and can be used to manage an application and IAP (In Application Purchase) item.

TAS-5 can be used to check the audit status of application or IAP item.

6.3.2.6 TAS-6

This interface is exposed by the Developer Support Component and can be used by Storefront to send the malicious report and to provide the application sale information.

6.4 Security Considerations

The TAS Enabler provides authentication, confidentiality and integrity protection for the operations between TAS Client, Storefront, Developer Support, Developer's Portal, Capability Resources Management and Capability Resource Provider.

This is a list of the security solutions between TAS components.

- **Use of Session-level Certificates (TLS, SSL)**
  If the network between the TAS components is not trusted (e.g. the Internet, a very large intranet, etc.), TLS/SSL can be used.

- **HTTP Authentication**
  Even though the most common form of HTTP authentication is the basic authentication (i.e. a userID/password pair), other forms of HTTP authentication (e.g. digest) is preferable. The major difference between this approach and the use of TLS/SSL is that the latter is stronger in scalability and confidence, while the former is weaker in these aspects.

- **OAuth 2.0**
  OAuth 2.0 can be used for authorization between TAS components.

- **A Combination of Technologies**
  Technologies could be combined. For example, the TAS Client can establish an anonymous TLS/SSL session, whereupon HTTP authentication could be used to authenticate the TAS Client.

7. Application State

The TAS Enabler SHALL support application state flow control. Detailed State Flow Control steps and its diagrams for developer supports, and Storefront are provided in 7.1, and 7.2, respectively.

7.1 Application State Flow Control for Developer Support

A state flow control diagram is depicted in Figure 1 for the Developer Support. There are at least 5 states in TAS application state transition, which are represented in solid ellipse in figure 2 including Offline, Online, Submitted, Audited, Tested and End.

A solid arrow indicates the procedures in TAS for developers application management. A triggering event to trigger the state transition is also noted on the arrow with the direction included.

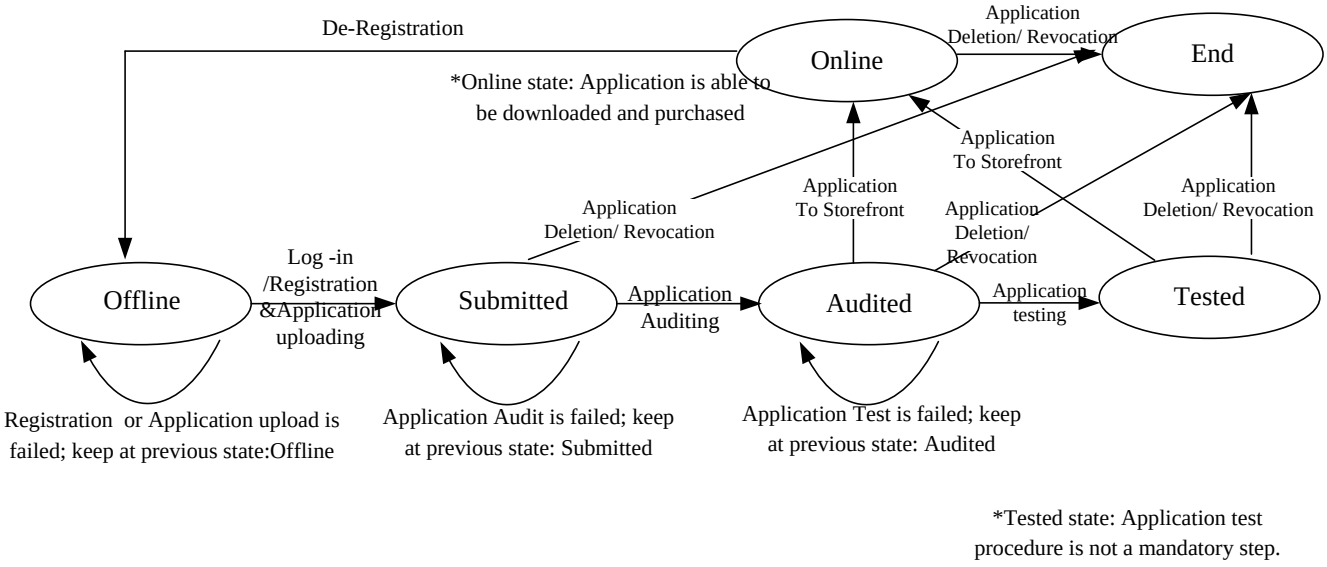

Figure 2: Application State Flow Control

Based on the Figure 1 above, the triggering events of each state transition are described respectively below:

- Offline→Submitted: When the developers have already successfully logged in and registered to TAS servers / TAS service providers, the developers are permitted to upload the applications developed and to access to resources. When developers start to upload an application, the procedure is mandated in Application upload flow as illustrated in Figure3. After the developers upload the application, application state successfully transits from Offline to Submitted.

  If registration or Application upload is failed, the application state keeps at previous state, Offline
- Submitted →Audited: After developers successfully upload the applications, they have to pass TAS service provider's audit procedure which is mandated in Application upload flow as illustrated in D1.1 Application Upload Flow. When to trigger the audit process and which information to audit depends on the Service Provider's policy.

  If Application Audit is failed, the application state keeps at previous state, Submitted.
- Audited→Tested: The developers could request internal testing of their submitted applications in Developers

Community, but this is not a mandatory step. If Application Test is failed, the application state keeps at previous state, Audited.

- Audited→Online: When the uploaded application by developers is successfully audited, the application can be published on the Storefront for TAS client's downloading and purchase. The TAS Enabler guarantees that only those Applications of online state could be downloaded.
- Tested→Online: When the uploaded application by developers is successfully audited and tested, the application can be published on the Storefront for TAS client's downloading and purchase. The TAS Enabler guarantees that only those Applications of online state could be downloaded.
- Online→End: When the uploaded application is deleted or has been through the application revocation, application state transits from Online to End.
- Online→Offline: When the developers de-register to the TAS servers / TAS service providers, application state transits from Online to Offline.
- Submitted→Offline: At Submitted state, the developers try to delete the application or make application revocation before this application is Online, application state transits from Submitted to End.
- Audited →Offline: At Audited state, the developers try to delete the application or make application revocation before this application is Online, application state transits from Audited to End.
- Tested →Offline: At Tested state, the developers try to delete the application or make application revocation before this application is Online, application state transits from Tested to End.

7.2 Application State Flow Control for Storefront

A state flow control diagram for Storefront is depicted in Figure 2. There are 3 states in TAS application state transition, which are represented in solid ellipse in Figure 2 including Offline, Online, and invalidated.

A solid arrow indicates the procedures executed in TAS. A triggering event to trigger the state transition is also noted on the arrow with the direction included.

Besides, the Storefront needs to keep the application state list of each application respectively.

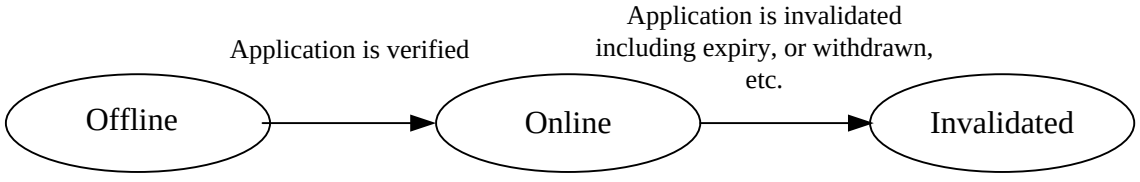

Figure 3: Application State Flow Control For Storefront

Based on the Figure 2 above, the triggering events of each state transition are described respectively below:

- Offline→Online: When the uploaded application by developers is successfully verified, and the Storefront is ready to publish the application on market. These applications of online state could be downloaded.
- Online→Invalidated: When the application is not allowed to be public on the Storefront, such as the application is expired, withdrawn by the Storefront, etc.

8. TAS Operations and Functions

8.1 Storefront Operations and Functions

8.1.1 Application Download

After trust is established between the Storefront and the TAS Client, the Storefront can send applications to the TAS Client when requested. The request from the TAS Client is defined as the AppDownloadRequest message in section 9.2.1.1, which consists of a list of application IDs. Upon receiving the message, the Storefront SHALL extract all applications IDs, and send applications indicated by them back to the TAS Client, via the AppDownloadResponse message defined in section 9.2.1.2. The Storefront SHOULD prepare as many applications as available.

If all the applications requested are packaged to the AppDownloadResponse message and sent back, the status code SHOULD be success. If not all the applications requested are sent back, the status code SHOULD NOT be success.

8.1.2 Application Transfer

Upon reception of the AppInfoNotify message (see Section 9.1.1) from the Developer Support, the Storefront Shall store description information of all the applications ,to identify what application(s) it intends to be transferred, the Storefront SHALL provide Application ID(s) from the AppInfoList and send AppTransferRequest message including the AppIDList and other parameters, as described in Section 9.1.1.1 to the Developer Support.

After receiving the AppTransferResponse message, the Storefront Shall store the related variants of all the applications.

8.1.3 Malicious Application

After trust is established between the Storefront and the TAS Client, the Storefront can send applications to the TAS Client when requested. The report from the TAS Client is defined as the Malicious Application Report message in section 9.2.2.1 which consists of the ID of the application to be reported as a malicious application, and might also include the comment which is the reason to report this Malicious Application, or suggestion to handle this Malicious Application. Upon receiving the message, the Storefront SHALL extract all applications IDs, and send Malicious Application Response defined in section 9.2.2.2 back to the TAS Client. The Malicious Application Response consists of the result which is the Status code or error information.

Besides, after trust is established between the Storefront and the Developer Support, the Storefront SHALL send Malicious Application Notification to notify the developer support when a list of malicious application regarding to the Developer support is ready. Malicious Application Notification consists of a list of application IDs to be notified. The list named as Commentlist SHALL include the Number of application IDs in this list, might include the ID of the application and the Reason to notify this Malicious Application, or suggestion to handle this Malicious Application.

8.1.4 Application Sorting

After trust is established between the Storefront and the TAS Client, the Storefront can send a list of sorted applications to the TAS Client when requested. The request from the TAS Client is defined as the Application Sorting Request message in section 9.2.10.1, which consists of criteria to apply on the Storefront for sorting. Upon receiving the message, the Storefront SHALL extract the criteria and send sorted applications list and corresponding application information back to the TAS Client, via the Application Sorting Response message defined in section 9.2.10.2.

If the applications sorted are packaged to the Application Sorting Response message and sent back, the status code SHOULD be success.

8.1.5 Application Search

After trust is established between the Storefront and the TAS Client, the Storefront can send searched applications based on attributes to the TAS Client when requested. The request from the TAS Client is defined as the Application Search Request message in section 9.2.11.1, which consists of attributes used by search. Two kinds of search is performed, one is “Quick Search” which means only one single attribute is applied in the textbox, while the other is “Advanced Search” which means

combination of several attributes is provided and applied in the textbox for the search. Quick Search and Advanced Search is mutually exclusively, which means only one kind of search can be performed at one time.

Upon receiving the message, the Storefront SHALL extract the attributes for search , send searched result of applications list back to the TAS Client, via the Application Search Response message defined in section 9.2.11.2.

If the applications searched are packaged to the Application Search Response message and sent back, the status code SHOULD be success.

8.1.6 Application State Transition Notification

After trust is established between the Storefront and the Developer Support, if some applications encounter state transition on the Storefront, such as from "offline" state to "online" state or from "online" state to "invalidated" state, the Storefront SHALL send a list of these application state transitions to the Developer Support in the Application State Transition Notification message defined in section 9.1.4.

8.1.7 Application State Management

After the application is transferred from the Developer Support to the Storefront, the State of the Application is "Offline". Then if the Storefront is going to publish the application on the market, the state will be shifted to "Online" automatically or by the administrator of the Storefront. And then, the state may be shifted to" Invalidated" by some reasons which are defined by the TAS service provider's policy, e.g. unsalable for a long time, expired and so on.

Each application in the Storefront SHALL be set with one state. Each state transition is caused by a reason. The Storefront SHALL record the reason of each state transition.

Only those Applications of online state in the Storefront can be displayed, purchased, downloaded.

8.1.8 Application Category Management

The Storefront SHALL provide the functions that support the administrator of the Storefront to manage category of the Storefront.

The management functions include but not limit to the following:

Define a list of possible Application Categories. The Application Category includes (not exhaustive): category name, category id, description element(s), etc.

Define a list of possible Application SubCategories. The Application SubCategory includes (not exhaustive): category name, category id, description element(s), etc.

Add/ Delete an Application Category

Add/Delete an Application SubCategory

Delete the Application from the Category when it is not appropriate.

8.1.9 Application Sales Report

After the application purchase process is finished successfully, The storefront stores the application purchase information, generates the sales report of this application, sends the report to the developer support via the TAS-6 interface.

Periodically, the storefront and the developer support do the settlement based on the statistics of those application sales reports.

8.1.10 Application Purchase Record Response

After trust is established between the Storefront and the TAS Client, the Storefront can send AppPurchaseRecord Response to the TAS Client when requested. The request from the TAS Client is defined as the AppPurchaseRecord Request message in section 9.2.17.1.

Upon receiving the message, the Storefront extracts the attributes for application ID, application name, and the purchase date for this purchase record summary if these attributes are available.

Otherwise, the complete application purchase record will be provided.

AppPurchaseRecord Response includes AppPurchaseInfoList which is the list of application(s) for the Purchase information. This list includes the number of applications in this list and its corresponding application purchase information such as application ID, application name, purchase date and Version.

8.1.11 Application Update

Upon receiving the AppUpdateNotify (see Section 9.1.3.2) message from the Developer Support, The Storefront SHOULD send an AppUpdateRequest (see Section 9.1.3.2) to the Developer Support. The Storefront SHALL include in the AppUpdateRequest all the applications the Storefront had selected from the AppUpdateNotify.

8.1.12 Application Feedback (Optional)

Upon receiving the AppFeedbackRequest message, the Storefront SHALL send AppFeedbackResponse message defined in section 9.1.3.1 to TAS Client. The comment of the application for that Storefront SHOULD be recorded accordingly.

If the application feedback transition message is received successfully, the result in AppFeedbackResponse message SHOULD be success. Otherwise, the result SHOULD NOT be success.

8.1.13 Application Download Status Response

Upon receiving the Application Download Status Report message defined in section 9.2.14.1, the Storefront SHALL send Application Download Status Response message defined in section 9.2.14.2 to the TAS Client. The Application Download Status of the application for that Storefront SHOULD be recorded accordingly.

If the Application Download Status Report message is received successfully, the result in Application Download Response message SHOULD be success. Otherwise, the result SHOULD NOT be success.

8.1.14 Application Install Status Report

Upon receiving the Application Install Status Report message defined in section 9.2.5.1, the Storefront SHALL send Application Install Status Response message defined in section 9.2.5.2. The Application Install Status of the application for that Storefront SHOULD be recorded accordingly.

If the Application Install Status Report message is received successfully, the result in Application Install Response message SHOULD be success. Otherwise, the result SHOULD NOT be success.

8.1.15 User Activation

After trust is established between the Storefront and the TAS Client, the Storefront can activate user when TAS Client requests. The request from the TAS Client is defined as the UserActivationRequest message in section 9.2.4.1, which consists of a UserID and a TerminalInfo. Upon receiving the message, the Storefront SHALL decide whether to activate the user or not, and send the result to the TAS Client via the UserActivationResponse message defined in section 9.2.4.2.

If the user can be activated, the status code SHOULD be success. If not, the status code SHOULD NOT be success.

8.1.16 User Deactivation

After trust is established between the Storefront and the TAS Client, the Storefront can deactivate user when TAS Client requests. The request from the TAS Client is defined as the DeactivationRequest message in section 9.2.12.1, which consists of a UserID and a User-Password (as optional). Upon receiving the message, the Storefront SHALL decide whether to deactivate the user or not, and send the result to the TAS Client via the DeactivationResponse message defined in section 9.2.12.2.

If a user password is needed when deactivate user, the Storefront should examine the password before the deactivation.

If the user can be deactivated, the status code SHOULD be success. If not, the status code SHOULD NOT be success.

8.1.17 User Account Information Check

After trust is established between the Storefront and the TAS Client, the Storefront can feedback the user account information when requested. The request from the TAS Client is defined as the UserAcInfCheckRequest message in section 9.2.6.1, which consists of a UserID and a UserPasswd (as optional). Upon receiving the message, the Storefront SHALL decide whether to feedback the user account information or not, and send the result to the TAS Client via the UserAcInfCheckResponse message defined in section 9.2.6.2.

The User Account information should consist of UserID, the status of User in TAS, the Description for whether User need password to Login, etc.

If a user password is needed, the Storefront should examine the password before feedback the user account information.

If the user account information can be feed backed, the status code SHOULD be success. If not, the status code SHOULD NOT be success.

8.1.18 User Account Information Modification

After trust is established between the Storefront and the TAS Client, the Storefront can modify the user account information when requested. The request from the TAS Client is defined as the AccountInfModifyRequest message in section 9.2.7.1. Upon receiving the message, the Storefront SHALL decide whether to modify the user account information, and send the result to the TAS Client via the AccountInfModifyResponse message defined in section 9.2.7.2.

If the user account information can be modified, the status code SHOULD be success. If not, the status code SHOULD NOT be success.

8.1.19 Application Refund

Upon receiving the AppRefundRequest message defined in section 9.2.13.1, the Storefront SHALL check and verify the user's billing of the application and send AppRefundResponse message defined in section 9.2.13.2. The Storefront SHALL handle the request of the refund accordingly.

If the refund is handled successfully, the result in AppRefundResponse message SHOULD be success. Otherwise, the result SHOULD NOT be success.

8.1.20 Feedback Reviews (Optional)

After trust is established between the Storefront and the TAS Client, After the TAS Client send FeedbackListRequest to the Storefront, the Storefront Shall send FeedbackList Response message defined in section 9.2.16.2 to the TAS Client.

How many Feedback Records will be included in the Response message, This number depends on the policy of TAS operator.

8.1.21 Application Gift (Optional)

After trust is established between the Storefront and the TAS Client, After the TAS Client send Application Gift Request to the Storefront, the Storefront Shall send Application Gift Response message defined in section 9.2.21.2 to the TAS Client.

Upon receiving the Application Gift Request message, the Storefront extracts the parameters for application ID, UserID, ReceiveUserID and GiftMessage. Within these parameters the Storefront form a gift notify message, and send to the ReceiveUser. According to which way to send this message, it depends on the policy of the TAS service provider e.g. short message or email. If send this notify message successfully, the Storefront Shall send Application Gift Response message with successful result to the TAS Client. Otherwise, with a fail result.

8.1.22 Application Recommendation

After trust is established between the Storefront and the TAS Client, after the TAS Client sends Application Recommendation Request to the Storefront, the Storefront SHALL send Application Recommendation Response message defined in section 9.2.22.2 to the TAS Client.

Upon receiving the Application Recommendation Request message, the Storefront extracts the parameters for Application ID, UserID, ReceiveUserID and Recommendation Message. Within these parameters the Storefront forms a Recommendation notify message, and sends to the ReceiveUser. The way to send this message, depends on the policy of the TAS service provider, e.g. Short Message, WAP Push or email. If sending this notification message successfully, the Storefront SHALL send Application Recommendation Response message with successful result to the TAS Client. Otherwise, with a fail result.

8.1.23 Gift Reception (Optional)

After trust is established between the Storefront and the TAS Client, the Storefront can send Giftlist Response to the TAS Client when requested. The Response message from the Storefront is defined as the GiftlistResponse message in section 9.2.23.2.

GiftlistResponse includes Giftlist which is the list of application(s) for the Gift information. This list includes the number of applications information in this list and its corresponding gift UserID, such as application ID, application name, Version, application detailed information, GiftUser.

8.1.24 Application Shopping Cart List Creation (Optional)

The Storefront is responsible for the management of application shopping cart list. The Storefront should support to create the shopping cart list both through the interface between the TAS Client and the Storefront, and through the browser way of TAS Client, which means the TAS Client could directly access the information on the Storefront.

(Note: it is suggested that the browser way of TAS Client should also be applied in the function of Application Sorting, Application Search, Application FeedbackList View and Application Favorites)

If the shopping cart list is created by the interface method, the operation is as follows: After trust is established between the Storefront and the TAS Client, the Storefront can create an application shopping cart list when requested. The request from the TAS Client is defined as the AppShoppingCartListCreation Request message in section 9.2.24.1, which consists of a UserID and an AppIDList structure. Upon receiving the message, the Storefront SHALL decide whether to create the application shopping cart list or not, and send the result to the TAS Client via the AppShoppingCartListCreation Response message defined in section 9.2.24.2.

If the application shopping cart list can be created, the status code SHOULD be success. If not, the status code SHOULD NOT be success.

8.1.25 Application Shopping Cart List Modification (Optional)

The Storefront is responsible for the management of application shopping cart list. The Storefront should support to modify the shopping cart list both through the interface between the TAS Client and the Storefront, and through the browser way of TAS Client, which means the TAS Client could directly access the information on the Storefront.

If the shopping cart list is modified by the interface method, the operation is as follows: After trust is established between the Storefront and the TAS Client, the Storefront can modify the application shopping cart list when requested. The request from the TAS Client is defined as the AppShoppingCartListModification Request message in section 9.2.25.1, which consists of a UserID and an AppShopCartModifyList structure. Upon receiving the message, the Storefront SHALL decide whether to modify the application shopping cart list or not, and send the result to the TAS Client via the AppShoppingCartListModification Response message defined in section 9.2.25.2.

If the application shopping cart list can be modified, the status code SHOULD be success. If not, the status code SHOULD NOT be success.

8.1.26 Application Shopping Cart List Inquiry (Optional)

The Storefront is responsible for the management of application shopping cart list. The Storefront should support to feedback the shopping cart list both through the interface between the TAS Client and the Storefront, and through the browser way of TAS Client, which means the TAS Client could directly access the information on the Storefront.

If the shopping cart list is obtained by the interface method, the operation is as follows: After trust is established between the Storefront and the TAS Client, the Storefront can feedback the application shopping cart list when requested. The request

from the TAS Client is defined as the AppShoppingCartListInquiry Request message in section 9.2.28.1, which consists of a UserID and a UserPasswd (as optional). Upon receiving the message, the Storefront SHALL decide whether to provide the application shopping cart list to the TAS Client, and send the result to the TAS Client via the AppShoppingCartListInquiry Response message defined in section 9.2.28.2.

If the application shopping cart list inquiry is successful, the status code SHOULD be success. The Storefront should provide the AppInfoList to the TAS Client, which consists of the list of application (s) put in the shopping cart. If not, the status code SHOULD NOT be success.

8.1.27 Application Begging

After trust is established between the Storefront and the TAS Client, the Storefront can send begging notify message to the TAS Client when requested. The request from the TAS Client is defined as the ApplicationBeggingRequest message in section 9.2.29.1. The TAS Client send Application Begging Request to the Storefront, the Storefront SHALL send Application Begging Response message defined in section 9.2.29.2 to the TAS Client.

Upon receiving the Application Begging Request message, the Storefront extracts the parameters for ApplicationID, UserID, GiverID and BeggingMessage. Within these parameters the Storefront form a begging notify message, and send to the GiverID. According to which way to send this message, it depends on the policy of the TAS service provider e.g. short message or email. If send this notify message successfully, the Storefront SHALL send Application Begging Response message with successful result to the TAS Client, but otherwise is fail result.

8.2 Developer Support Operations and Functions

8.2.1 Application Transfer

If some new applications in the Developer Support are ready, the Developer Support SHALL send an AppInfoNotify message to the Storefront including the DevSupportID and AppInfoList parameters, as described in Section 9.1.1.1.

Upon receiving the AppTransferRequest message from the Storefront, The Developer Support SHALL send an AppTransferResponse (see Section 9.1.1.3) to the Storefront. The Developer Support SHALL include in the AppTransferResponse all the applications the Storefront had selected from the AppTransferRequest.

If all the applications requested sent in the AppTransferResponse message are transferred successfully, the status code SHOULD be success. If not all the applications requested are transferred successfully, the status code SHOULD NOT be success.

8.2.2 Developer Registration

A Developer registers to the TAS Enabler as either an individual developer or an enterprise developer, by submitting a registration request to the Developer Support component.

The request of an individual developer registration include information of category, name, password, personal ID number, phone number, email address, bank account information, level and affiliated enterprise.

The request of an enterprise developer registration includes information of category, name, password, address, license number, amount of registered capital, bank of registered capital, bank account, web site address, contact name, contact email address, contact phone number and level.

Upon receiving the request, the Developer Support checks all the information. If it is authentic and complete, the Developer Support sends a “success” status code in a response message; otherwise a error status code is responded.

8.2.3 Application Upload

After successfully registered and contracted, a developer can submit applications to the TAS Enabler by sending an application upload message to the Developer Support component. This message consists of two parts: the application and its related information. The related information of an application includes Application name, type, keywords, proposed price, publish date, expiration date and charging option, etc.

Upon receiving the message, the Developer Support SHALL responds with a “success” status code before checking the application and its related information in it. This response only indicates that the application and its related information are

successfully received. Meanwhile, the Developer Support stores the application and its related information, and set its state as "Submitted".

8.2.4 Malicious Application

After trust is established between the Storefront and the Developer Support, the Developer Support can send Malicious Application Notification Response to the Storefront when receiving Malicious Application Notification message defined in section 9.6.1.1. If the procedure to notify the malicious application to the Developer Support is successful, the status code in Malicious Application Notification Response message SHOULD be success. Otherwise, the status code SHOULD NOT be success, and the status code and error information SHOULD be included.

8.2.5 Application Deletion

Upon receiving the Application Deletion Request message, the Developer Support SHALL extract the applications ID, and group the applications ID based on same reason for deletion , and send the list of application IDs to be deleted to the Storefront via the Application Deletion Response message defined in section 9.1.2.2.

If all the applications requested for deletion are successfully processed, the Developer Support sends the Application Deletion

Response message back to the developer's Portal, the status code SHOULD be success. If not, the status code SHOULD NOT be success, and the status code and error information SHOULD be included.

8.2.6 Application Audit and Test

After developer successfully uploads the application, the state of the application is shifted to "Submitted". Developer Support SHALL perform audit processing which is mandated in Application upload flow as illustrated in D1.1 "Application Upload Flow". When to trigger the audit process and which information to audit are based on the Service Provider's policy. After the auditing process is completed successfully, the state of the application SHALL be shifted to "Audited". If the auditing process is not passed, the state of the application SHALL not be changed and the reason SHOULD be recorded.

When the state of the application is shifted from "Submitted" to "Audited", Developer Support SHOULD perform test processing. When to trigger the test process and how to test are based on the Service Provider's policy. After a successful test procedure, the state of the application SHALL be shifted to "Tested". If the test process is not passed, the state of the application SHALL not be changed and the reason SHOULD be recorded.

8.2.7 Application State Transition Notification

Upon receiving the Application State Transition Notification message, the Developer Support SHALL send Application State Transition Response message defined in section 9.6.3.2. The state of the applications for that Storefront SHOULD be recorded accordingly.

If the state transition notification messages is received successfully, the status in Application State Transition Response message SHOULD be success. Otherwise, the status SHOULD NOT be success, Status code and error information SHOULD be included.

8.2.8 Application State Management

After the developer sends the request to upload application, the Developer Support generates an application ID for the developer's application.

The state will be shifted automatically or by the administrator of the Developer Support or the Developer. Each application in the Developer Support SHALL be set with one state. Each state transition is caused by a reason. The Developer Support SHALL record the reason of each state transition.

The Developer could view the state of his application by visiting the Developer's portal.

8.2.9 Developer State Management

After the Developer Support checks the information of the developer registration successfully, the state of the developer is "registered". The developer SHALL be able to access the resources in the Developer Support.

The Developer SHOULD contract with the Service Provider before he could submit Applications to it. How to sign the contract between the Service Provider and the Developer is defined by the TAS service provider's policy and it is out of scope of TAS Enabler. And each contract is valid in a defined period. During the period, the state of the Developer is "contracted", Otherwise, it is "registered".

If the Developer deregisters in the Developer Support, the Developer Support will delete the information of this Developer.

Only the Developer with "contracted" state could submit Applications to the Developer Support.

The Developer can view his state from the Developer's portal.

8.2.10 Application Status Check Response

After trust is established between the Developer's Portal and the Developer Support, upon receiving the Application Status Check Request message defined in Section 9.5.2.1, the Developer Support SHALL send the Application Status Check Response. The Developer Support extracts the applications ID, and sends the current audited status and its corresponding fail cause for not being able to be online of the application requested in the Application Status Check Request message defined in section 9.5.2.1.

8.2.11 Application Update

When some audited Applications in the Developer Support are updated, the Developer Support can send an AppUpdateNotify message to the Storefront to inform the updated Applications including the DevSupportID, StoreID and AppIDList parameters, as described in Section 9.1.3.1.

Upon reception of AppUpdateRequest from the Storefront, the Developer Support SHALL send an AppUpdateResponse (see Section 9.1.3.3.) to the Storefront to transfer the updated Applications.

8.3 Capability Resources Management Operations and Functions

This function is postponed to the future release.

8.4 TAS Client Operations and Functions

8.4.1 Application Download

After trust is established between the Storefront and the TAS Client, the TAS Client can request applications from the Storefront. To identify what application(s) it intends to download, the TAS Client SHOULD provide the ID(s) of all of them. If there is no ID in the request, the Storefront selects applications based on device information and user information. The device information and user information can be obtained in the UserActivationRequest message. The applications requested are expected to be sent back in an AppDownloadResponse message. Upon receiving this message, the TAS Client SHALL extract all applications in it. If the Content element of the Application element is present, the TAS Client MAY NOT parse the Content-Address element of the Application element; If the Content element of the Application element is not present, the TAS Client SHALL parse the Content-Address element of the Application element.

Whether and how the TAS Client makes use of the Content-Address element to acquire the application content is out of the TAS specifications scope.

If the status code of the AppDownloadResponse message is not success, the TAS Client SHOULD feedback the status code to the User.

8.4.2 Malicious Application

After trust is established between the Storefront and the TAS Client, the TAS Client can send a Malicious Application Report to the Storefront to provide certain information of malicious applications.

The TAS Client SHALL provide the ID of this Application, and should provide comment why this application is reported as malicious if possible. TAS Client is expected to receive Malicious Application Response message defined in section 9.6.1.2 after sending the Malicious Application Report. Upon receiving this message, the TAS Client is aware whether this reporting procedure is successful or not. An error code or information will be provided if this procedure fails.

8.4.3 Application Sorting (Optional)

After trust is established between the Storefront and the TAS Client, the TAS Client can request to get sorted list of applications based on criteria applied on the Storefront. The TAS Client SHALL provide the Criteria used by sorting. The list of sorted applications is expected to be sent back in an Application Sorting Response message defined in section 9.2.10.2. Upon receiving this message, the TAS Client SHALL extract all applications in it which consists of number of application inside the list and the application information.

8.4.4 Application Search (Optional)

After trust is established between the Storefront and the TAS Client, the TAS Client can request searched applications list based on certain attributes to the Storefront.

The TAS Client SHALL provide the attribute used by search. Two kinds of attributes are QuickSeach Attribure and AdvancedSearch Attribute. QuickSeach Attribure is only with a single attribute, while AdvancedSeach Attribute is a combination of several attributes. It might consist of the minimum price, the maximum price, the name of the Developer and also the application type. The application type is a Enumeration type with possible values:0,1,2,3, 4,5 ,6 corresponding to games, contents, tools, Entertainments, Books, Finance, Sports, News, others.

The list of search applications is expected to be sent back in a Application Search Response message defined in section 9.2.11.2. Upon receiving this message, the TAS Client SHALL extract all applications in it which consists of number of application inside the list and the application information.

8.4.5 Application Deletion

After trust is established between the Developer's Portal and the Developer Support, the Developer's Portal can send applications to the Developer Support to request for deleting an application. The request from the Developer's Portal is defined as the Application Deletion Request message in section 9.1.2.1, which consists of the identification of the application to be deleted and the reason why this application is deleted.

8.4.6 Application Purchase Record Request

After trust is established between the Storefront and the TAS Client, the TAS Client can send AppPurchaseRecord Request defined in section 9.2.17.1 to the Storefront to acquire the application purchase information.

The TAS Client could provide the attributes used for this application purchase record, such as Start date, End Date, application ID, and application name.

8.4.7 Application Feedback

TAS Client can provide the users' feedback of their experience on using the applications by submitting AppFeedbackRequest to the Storefront. The request of the AppFeedbackRequest includes UserID, ApplicationID, Rate and Comment information. The result of the application feedback is expected to be sent back in an AppFeedbackResponse message defined in Section 9.2.3.2.

8.4.8 Application Download Status Report

After the TAS Client had requested application download from the Storefront, the TAS Client can send the Application Download Status Report message to the Storefront. The request of the Application Download Status Report message includes UserID, ApplicationID and AppDownloadResult information.

If the application is downloaded successfully, the AppDownloadStatus in Application Download Status Report message SHOULD be success. Otherwise, the AppDownloadStatus SHOULD NOT be success, AppDownloadStatus and description SHOULD be included.

The result of application download status is expected to be sent back in an Application Download Status Response message defined in Section 9.2.14.2.

8.4.9 Application Install Status Report

After the TAS Client had downloaded the application from the Storefront successfully and finished installed, the TAS Client can send the Application Install Status Report message to the Storefront to inform the status of installing the application. The request of the Application Install Status Report message includes UserID, ApplicationID and AppInsResult information.

If the application is installed successfully, the AppInsStatus in Application Install Status Report message SHOULD be success. Otherwise, the AppInsStatus SHOULD NOT be success, AppInsStatus code and description SHOULD be included.

The result of the application Download is expected to be sent back in an Application Install Status Response message defined in Section 9.2.5.2.

8.4.10 User Activation

After trust is established between the Storefront and the TAS Client, the TAS Client can request the Storefront to activate user. To identify which user it intends to activate, the TAS Client SHALL provide the UserID in the request. The request from the TAS Client is defined as the UserActivationRequest message in section 9.2.4.1, which consists of a ClientID, UserID and a TerminalInfo. The TerminalInfo consists of the User's Terminal information, which includes the Hardware, the Operating System Type, etc. The Storefront SHOULD store the user's information and the user's terminal information associated with the ClientID, and use this information to select applications when it receives application download requests or application search requests.

The user activation response is expected to be sent back in a UserActivationResponse message. Upon receiving this message, the user is activated if the result of the UserActivationResponse is success. If the status code of the UserActivationResponse message is not success, the TAS Client SHOULD feedback the status code to the User.

8.4.11 User Deactivation

After trust is established between the Storefront and the TAS Client, the TAS Client can request the Storefront to deactivate user. To identify which user it intends to deactivate, the TAS Client SHALL provide the UserID in the request. The TAS Client SHALL also provide the User- Password in the request if needed. The request from the TAS Client is defined as the DeactivationRequest message in section 9.2.12.1, which consists of a UserID and a User-Password (as optional).

The user activation response is expected to be sent back in a DeactivationResponse message. Upon receiving this message, the user is deactivated if the result of the DeactivationResponse is success. If the status code of the DeactivationResponse message is not success, the user is not deactivated.

8.4.12 User Account Information Check

After trust is established between the Storefront and the TAS Client, the TAS Client can request User Account Information Check from the Storefront. To identify which user information it intends to check, the TAS Client SHALL provide the UserID in the request. The TAS Client SHALL also provide the User- Password in the request if needed. The request from the TAS Client is defined as the UserAcInfCheckRequest message in section 9.2.6.1, which consists of a UserID and a UserPasswd (as optional).

The user account information check request is expected to be sent back in a UserAcInfCheckResponse message. Upon receiving this message, the TAS Client SHALL show the user account information to the user if the result of the

UserAcInfCheckResponse is success. If the status code of the UserAcInfCheckResponse message is not success, the TAS Client SHOULD not show the user account information to the user.

8.4.13 User Account Information Modification

After trust is established between the Storefront and the TAS Client, the TAS Client can request User Account Information Modification to the Storefront. To identify which user account information it intends to modify, the TAS Client SHALL provide UserAccountInf, which includes the UserID in the request. The request from the TAS Client is defined as the AccountInfModifyRequest message in section 9.2.7.1, which consists of a UserAccountInf (the User Account information structure).

The user account information modification request is expected to be sent back in an AccountInfModifyResponse message. Upon receiving this message, the TAS Client SHALL show the user the account information modification is accomplished if the result of the AccountInfModifyResponse is success. If the status code of the AccountInfModifyResponse message is not success, the TAS Client SHOULD notify the user the modification is not successful.

8.4.14 Application Refund

After the TAS Client had finished purchasing the application, the TAS Client can send the AppRefundRequest message to the Storefront. The request of the Application Refund Request message include UserID, UserPasswd, RefundAppID, Price and RefundReason information.

The Result status of the AppRefund of the application Download is expected to be sent back in an Application Install Status Response message defined in Section 9.2.13.2.

The reasons of refund include but not limit to the following:

The application can not be downloaded successfully

The application can not be installed successfully

The application has some bugs and can not be used well

The user’s billing of the application is wrong

8.4.15 Feedback Reviews

After trust is established between the Storefront and the TAS Client, after the detail information of the ApplicationID is displayed, the TAS Client can send FeedbackList Request defined in section 9.2.16.1 to the Storefront to acquire the application Feedback list information.

Upon receiving the response message defined in section 9.2.16.2 from the Storefront, the TAS Client SHALL extract all the Rates, Dates, Comments, UserIDs and Versions in the message package, and displayed them to the Users.

8.4.16 Application Gift (Optional)

After trust is established between the Storefront and the TAS Client, the TAS Client can send Application Gift Request message defined in section 9.2.21.1 to the Storefront to gift receiveuser application.

Upon receiving the response message defined in section 9.2.21.2 from the Storefront, the TAS Client SHALL extract the application gift result (success or fail), and displayed it to the Users.

8.4.17 Application Recommendation

After trust is established between the Storefront and the TAS Client, the TAS Client can send Application Recommendation Request message defined in section 9.2.22.1 to the Storefront to send application recommendation to the receiveuser.

Upon receiving the response message defined in section 9.2.22.2 from the Storefront, the TAS Client SHALL extract the application recommendation result (success or fail), and displayed it to the Users.

8.4.18 Gift Reception (Optional)

After trust is established between the Storefront and the TAS Client, the TAS Client can send Giftlist Request message defined in section 9.2.23.1 to the Storefront to acquire the application Gift list information.

Upon receiving the Giftlist Response message defined in section 9.2.23.2 from the Storefront, the TAS Client SHALL extract the application gift list information, and displayed them to the Users.

After displaying, the TAS Client can send the application download request message defined in section 9.2.1.1 to download the gift application.

8.4.19 Application Shopping Cart List Creation (Optional)

The TAS Client should support to create Application Shopping Cart List both by an interface over TAS-2 and by browser way.

If the shopping cart list is created by the interface method, the operation is as follows: After trust is established between the Storefront and the TAS Client, the TAS Client can create application shopping cart lists on the Storefront. To identify which user the application shopping cart list belongs to, the TAS Client SHALL provide the UserID in the request. The TAS Client SHALL also provide the AppIDList in the request, which refers to the list of application ID(s) put in the shopping cart. The request from the TAS Client is defined as the AppShoppingCartListCreation Request message in section 9.2.24.1, which consists of a UserID and an AppIDList structure.

Upon receiving this message, the TAS Client is aware whether this shopping cart list creation is successful or not. An error code or information will be provided if this procedure fails.

8.4.20 Application Shopping Cart List Modification (Optional)

The TAS Client should support to modify Application Shopping Cart List both by an interface over TAS-2 and by browser way.

If the shopping cart list is modified by the interface method, the operation is as follows: After trust is established between the Storefront and the TAS Client, the TAS Client can modify the application shopping cart list from the Storefront. To identify which user the application shopping cart list belongs to, the TAS Client SHALL provide the UserID in the request. The TAS Client SHALL also provide an AppShopCartModifyList structure, which shows the modification(s) of the shopping cart list.

The request from the TAS Client is defined as the AppShoppingCartListModification Request message in section 9.2.25.1, which consists of a UserID and an AppShopCartModifyList structure.

Upon receiving this message, the TAS Client is aware whether this shopping cart list modification is successful or not. An error code or information will be provided if this procedure fails.

8.4.21 Application Shopping Cart List Inquiry (Optional)

The TAS Client should support to obtain the Application Shopping Cart List both by an interface over TAS-2 and by browser way.

If the shopping cart list is obtained by the interface method, the operation is as follows: After trust is established between the Storefront and the TAS Client, the TAS Client can inquire the application shopping cart list from the Storefront. To identify which user the application shopping cart list belongs to, the TAS Client SHALL provide the UserID in the request. The TAS Client SHALL also provide a User-Password if needed.

The request from the TAS Client is defined as the AppShoppingCartListInquiry Request message in section 9.2.28.1, which consists of a UserID and a User-Password (as optional)

Upon receiving this message, the TAS Client is aware whether this shopping cart list inquiry is successful or not. An error code or information will be provided if this procedure fails.

8.4.22 Application Begging (Optional)

After trust is established between the Storefront and the TAS Client, the TAS Client can send Application Begging Request message defined in section 9.2.29.1 to the Storefront for begging.

Upon receiving the response message defined in section 9.2.29.2 from the Storefront, the TAS Client SHALL extract the application begging result (success or fail), and displayed it to the Users.

8.5 Developer's Portal Operations and Functions

8.5.1 Application Status Check Request

After trust is established between the Developer's Portal and the Developer Support, the Developer's Portal can send Application Status Check Request message defined in Section 9.5.2.1. to query the current audited status for the Application developed. The request from the Developer's Portal is defined as the Application Status Check Request message in section 9.1.2.1, which consists of the identification of the application to be queried

8.6 Capability Resources Provider Operations and Functions

This function is postponed to the future release.

9. Interface Descriptions

9.1 TAS-1

9.1.1 Application Transfer

Audited Applications can be transferred from the Developer Support to the Storefront. While there may exist different ways to do that, one of them follows the procedure below:

- The Developer Support notifies the Storefront that some applications are ready;
- The Storefront selects Applications from the list and makes a transfer request to the Storefront;
- Then the requested Applications are sent back within a response message.

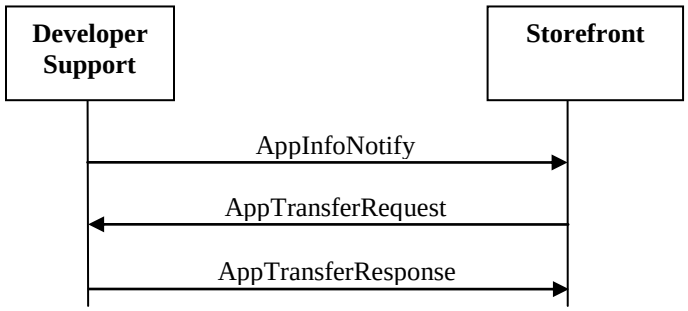

Figure 4: Application Transfer messages exchange

9.1.1.1 AppInfoNotify

Name	Cardinality	Data Type	Description
DevSupportID	1	String	The ID of the Developer Support which sends this message
StoreID	1	String	The ID of the Store which the application transfer message sent to.
AppInfoList	1	Structure	List of information of application(s).

Table 1: AppInfoNotify message

9.1.1.2 AppTransferRequest

Name	Cardinality	Data Type	Description
AppIDList	1	Structure	List of application ID(s) to be transferred

Table 2: AppTransferRequest message

9.1.1.3 AppTransferResponse

Name	Cardinality	Data Type	Description
AppList	0..1	Structure	List of application(s)
StatusCode	1	Integer	Status code

Table 3: AppTransferResponse message

9.1.2 Application Deletion

This operation enables the TAS Developers to delete its application on the Storefront.
It consists of two messages: An Application Deletion Request message from the Developer Support to the Support, and an Application Deletion Response message from the Storefront to the Developer Support.

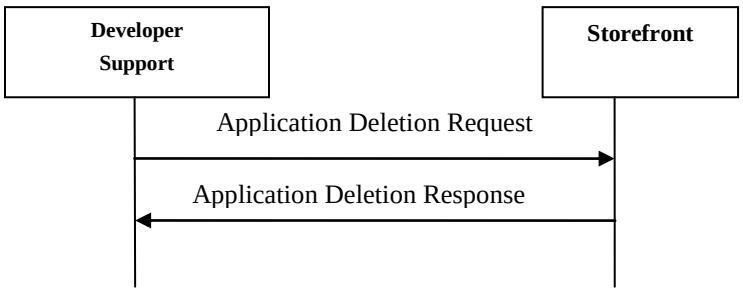

Figure 5: Application Deletion messages exchange

9.1.2.1 Application Deletion Request

Name	Cardinality	Data Type	Description
Developer-ID	1	String	The ID of the TAS Developer.
AppIDList	1	Structure	List of application ID(s) to be deleted
Reason	0..1	String	reason to delete this application

Table 4: Application Deletion Request message

9.1.2.2 Application Deletion Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the deletion Result.

Table 5: Application Deletion Response message

9.1.3 Application Update

Audited Applications can be updated from the Developer Support to the Storefront. While there may exist different ways to do that, one of them follows the procedure below:

- The Developer Support notifies the Storefront that some applications have been updated;
- The Storefront selects Applications from the list and makes a update request to the Storefront;
- Then the requested Applications are sent back within a response message.

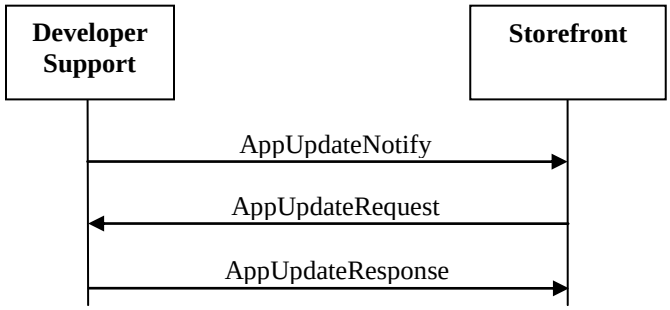

Figure 6: Application Update messages exchange

9.1.3.1 AppUpdateNotify

Name	Cardinality	Data Type	Description
DevSupportID	1	String	The ID of the Developer Support which sends this message
AppInfoList	1	Structure	List of information of application(s).

Table 6: AppInfoNotify message

9.1.3.2 AppUpdateRequest

Name	Cardinality	Data Type	Description
StoreID	1	String	The ID of the Storefront
AppIDList	1	Structure	List of application ID(s) to be transferred

Table 7: AppUpdateRequest message

9.1.3.3 AppUpdateResponse

Name	Cardinality	Data Type	Description
AppList	0..1	Structure	List of application(s)
Result	1	Structure	Result of the message

Table 8: AppUpdateResponse message

9.1.4 Application State transition Notification

This operation enables the Developer Support to send Application State Transition Notification to the Storefront to provide information on the status change of application on the Developer Support components

It consists of two messages: An Application State Transition Notification message from the Developer Support to the Storefront, and Application State Transition Response message from the Storefront to the Developer Support.

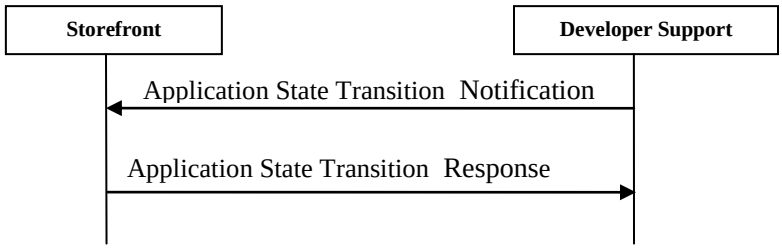

Figure 7: Application state transition Notification messages exchange

9.1.4.1 Application State Transition Notification

Name	Cardinality	Data Type	Description
DeveloperID	1	String	The ID of the Developer Support
SatteTransAppCount	1	Integer	Number of Application on the Developer Support which encounter state transition in this list
StateTransAppList	0..N	Structure	List of application ID(s) to be notified.

Table 9: Application State Transition Notification message

Name	Cardinality	Data Type	Description
AppID	1	String	The ID of the application which encounters state transition
StateTransTypeDeveloper	1	Integer	The description of the application state transition on Developer Support Component
statetransReason	0..1	String	Reason of the state transition

Table 10: StateTransAppList structure

Enumeration	Description
0	OnlineEnd : The state transition is from “Online” state to “End” state
1	OnlineOffline :The state transition is from “Online” state to Offline” state

Table 11: Statetranstype Enumeration

9.1.4.2 Application Sate Transition Response

Name	Cardinality	Data Type	Description
result	1	Structure	Status code and error information

Table 12: application state transition Response Message

9.1.5 IAP synchronize (Optional)

This operation enables the TAS Developer Support to synchronize its IAP item to the Storefront according to the synchronize policy made by the Developer Support.

It consists of two messages: An IAP item synchronize Request message from the Developer Support to the Storefront, and an IAP item synchronize Response message from the Storefront to the Developer Support.

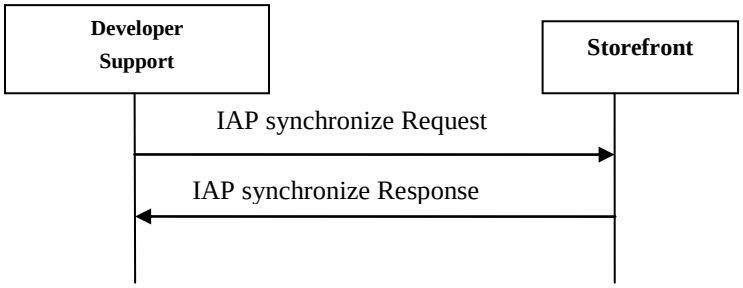

Figure 8: IAP synchronize messages exchange

9.1.5.1 IAP synchronize Request

Name	Cardinality	Data Type	Description
IAPinfo	1	Structure	The information of the IAP item

Table 13: IAP synchronize Request message

9.1.5.2 IAP synchronize Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the synchronize Result.

Table 14: IAP synchronize Response message

9.1.6 App type synchronize

This operation enables the TAS Developer Support to synchronize its application type to the Storefront according to the synchronize policy made by the Developer Support.

It consists of two messages: An App type synchronize Request message from the Developer Support to the Storefront, and an App type synchronize Response message from the Storefront to the Developer Support.

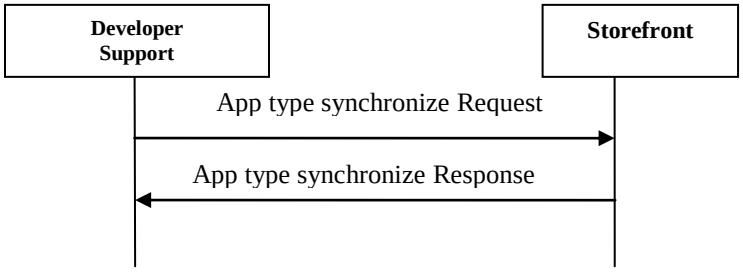

Figure 9: App type synchronize messages exchange

9.1.6.1 Apptype synchronize Request

Name	Cardinality	Data Type	Description
AppTypeList	1	structure	The apptype list of the Developer Support

Table 15: apptype synchronize Request message

9.1.6.2 Apptype synchronize Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the synchronize Result.

Table 16: apptype synchronize Response message

9.2 TAS-2

9.2.1 Application Download

This operation enables the TAS Client to download application(s) from the Storefront. It consists of two messages: An AppDownloadRequest message from the TAS Client to the Storefront, and an AppDownloadResponse message from the Storefront to the TAS Client.

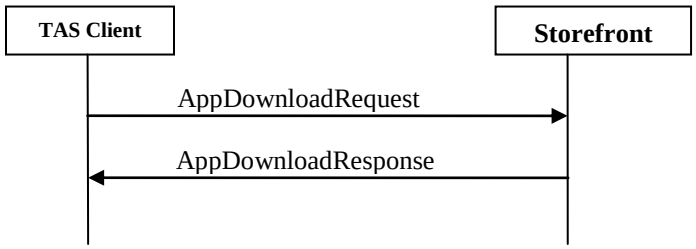

Figure 10: Application Download messages exchange

9.2.1.1 AppDownloadRequest

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
AppIDList	1	Structure	List of application ID(s) to be downloaded

Table 17: AppDownloadRequest message

9.2.1.2 AppDownloadResponse

Name	Cardinality	Data Type	Description
AppInfoList	0..1	Structure	List of application(s)
StatusCode	1	Integer	Status code

Table 18: AppDownloadResponse message

Name	Cardinality	Data Type	Description
AppCount	1	Integer	Number of applications in this list
Application	0..N	Structure	The application

Table 19: AppList structure

Name	Cardinality	Data Type	Description
Application-ID	1	String	The ID of the application
ApplicationName	1	String	The name of the application
Content-Address	1, Conditional	URI	The URI of the application content. Condition: This parameter is mutually exclusive with the content parameter.
Content	1, Conditional	Binary	The content of the application. Condition: This parameter is mutually exclusive with the content-address parameter.

Table 20: Application structure

9.2.2 Malicious Application Report

This operation enables the TAS Client to send Malicious Application Report to the Storefront, to provide information on Malicious applications they have found.

It consists of two messages: A Malicious Application Report message from the TAS Client to the Storefront, and a Malicious Application Response message from the Storefront to the TAS Client.

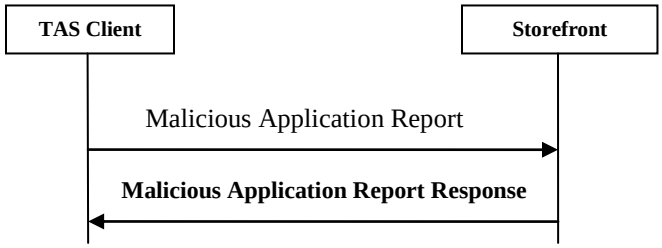

Figure 11: Malicious Application Report messages exchange

9.2.2.1 Malicious Application Report

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
ApplicationID	1	String	The ID of the application
Comment	0..1	String	reason to Report this Malicious Application, or suggestion to handle this Malicious Application

Table 21: Malicious Application Report message

9.2.2.2 Malicious Application Response

Name	Cardinality	Data Type	Description
result	1	Structure	Status code and error information

Table 22: Malicious Application Response message

9.2.3 Application Feedback (Optional)

This operation enables the User to submit application feedback to the Storefront. It consists of two messages: An AppFeedbackRequest message from the TAS Client to the Storefront, and an AppFeedbackResponse message from the Storefront to the TAS Client.

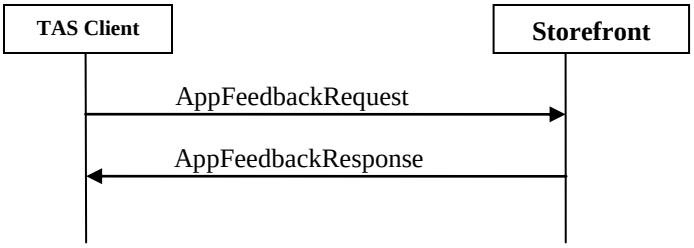

Figure 12: Application Feedback messages exchange

9.2.3.1 AppFeedbackRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
ApplicationID	1	String	The ID of the application
Rate	1	Integer	The Rate of the application feedback
Comment	1	String	The Comment content of the application feedback

Table 23: AppFeedbackRequest message

9.2.3.2 AppFeedbackResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the application feedback

Table 24: AppFeedbackResponse message

9.2.4 User Activation

This operation enables the TAS Client to Activate User in the Storefront. It consists of two messages: An UserActivationRequest message from the TAS Client to the Storefront, and an UserActivationResponse message from the Storefront to the TAS Client.

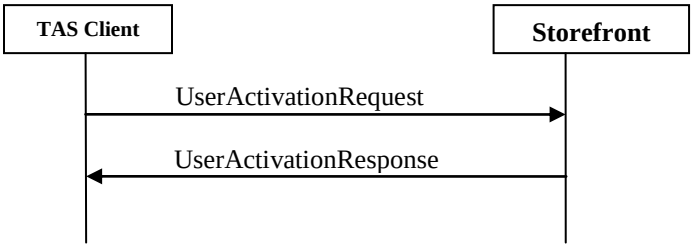

Figure 13: User Activation messages exchange

9.2.4.1 UserActivationRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
ClientID	1	String	The ID of the TAS Client
TerminalInfo	1	Structure	The Information of the User's Terminal (Hardware,Operating System Type&Version)

Table 25: UserActivationRequest message

9.2.4.2 UserActivationResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the UserActivation

Table 26: UserActivationResponse message

9.2.5 Application Install Status Report

This operation enables the TAS Client to report the application(s)'s status of install/uninstall to the Storefront. It consists of two messages: An AppInsStatusReport message from the TAS Client to the Storefront, and an AppInsStatusReportResponse message from the Storefront to the TAS Client.

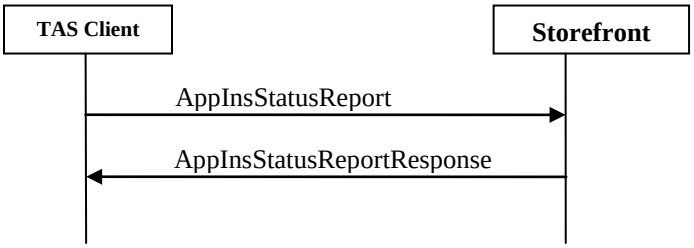

Figure 14: Application Install Status Report messages exchange

9.2.5.1 AppInsStatusReport

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS user
ApplicationID	1	String	The ID of the aplication
AppInsResults	1	Structure	Application install result

Table 27: AppInsStatusReportRequest message

Name	Cardinality	Data Type	Description
AppInsStatus	1	Enumeration	The installation status
Description	0..1	String	The detailed information about the status code

Table 28: AppInsResult Structure

9.2.5.2 AppInsStatusReportResponse

Name	Cardinality	Data Type	Description
Result	0..n	Result	Result of application install status

Table 29: AppInsStatusReportResponse message

9.2.6 User Account Information Check

This operation enables the User to check user account information in the Storefront. It consists of two messages: An UserAcInfCheckRequest message from the TAS Client to the Storefront, and an UserAcInfCheckResponse message from the Storefront to the TAS Client.

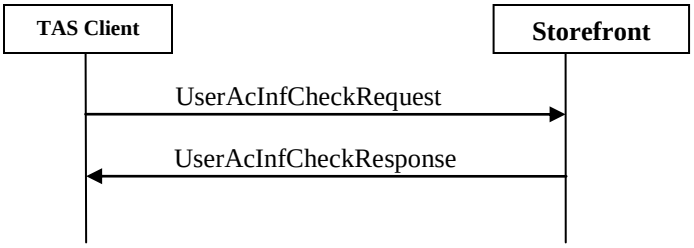

Figure 15: User Account Information Check messages exchange

9.2.6.1 UserAcInfRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserPasswd	0..1	String	The password of User

Table 30: UserAcInfRequest message

9.2.6.2 UserAcInfResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of AccoutInfo Check Result
UserAccountInf	0..1	Structure	User Account information structure Condition: This parameter is only with the Value of the Status of Result parameter is Success.

Table 31: UserAcInfResponse message

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserStatus	1	Integer	The status of User in TAS(e.g. active, inactive)
PaymentAccount	0..1	Structure	The User's Paid-Account detail description.
PaymentType	0..1	Enumeration	The Payment-type of TAS User (e.g. post-paid, prepaid). This parameter SHALL not be present in AccountInfModifyRequest.
PasswdLogin	1	Boolean	The Description for whether User need Passwd to Login.
PasswdPurchase	1	Boolean	The Description for whether User need Passwd to Purchase

Table 32: UserAccountInf Structure

Name	Cardinality	Data Type	Description
TelecomAccount	0..1	String	Telecom Account Description.
BankCard	0..1	String	The User's Bank Card Information.
ThirdAccount	0..1	String	The User's Third Account Information (e.g. PayPal\zhifubao).

Table 33: Payment-Account Structure

9.2.7 User Account information modification

This operation enables the User to modify Account Binding information in the Storefront. It consists of two messages: An AccountInfModifyRequest message from the TAS Client to the Storefront, and an AccountInfModifyResponse message from the Storefront to the TAS Client.

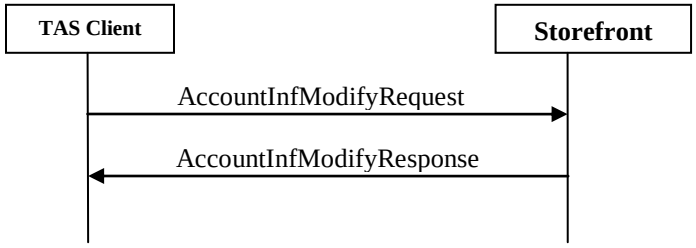

Figure 16: User Account information modification messages exchange

9.2.7.1 AccountInfModifyRequest

Name	Cardinality	Data Type	Description
UserAccountInf	1	Structure	User Account information structure

Table 34: AccountInfModifyRequest message

9.2.7.2 AccountInfModifyResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The result status code of the User Account information modification

Table 35: AccountInfModifyResponse message

Note:

Status code of Result

Invalid Account

Account type not supported

9.2.8 Application Purchase

This operation enables the User to Purchase Application in the Storefront. It consists of two messages: An AppPurchaseRequest message from the TAS Client to the Storefront, and an AppPurchaseResponse message from the Storefront to the TAS Client.

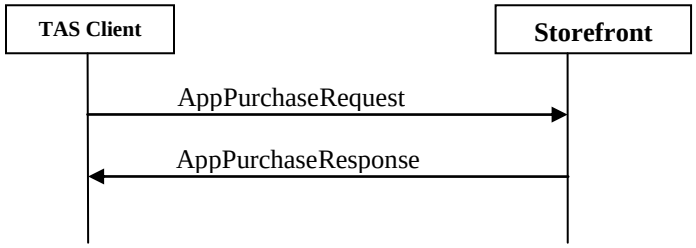

Figure 17: Application Purchase messages exchange

9.2.8.1 AppPurchaseRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserPasswd	0..1	String	The Passwd of User
AppIDList	1	Structure	List of Applications to be purchased

Table 36: AppPurchaseRequest message

9.2.8.2 AppPurchaseResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of the AppPurchase

Table 37: AppPurchaseResponse message

Note:

Code Description,

no pay account binding

NSF check, Insufficient fund

9.2.9 Application Update

This operation enables the User to Update Application from the Storefront. It consists of six messages: An AppUpdateCheckRequest message from the TAS Client to the Storefront, and an AppUpdateCheckResponse message from the Storefront to the TAS Client.; An AppUpdateDetailRequest message from the TAS Client to the Storefront, and an AppUpdateDetailResponse message from the Storefront to the TAS Client; An AppUpdateRequest message from the TAS Client to the Storefront, and an AppUpdateResponse message from the Storefront to the TAS Client.

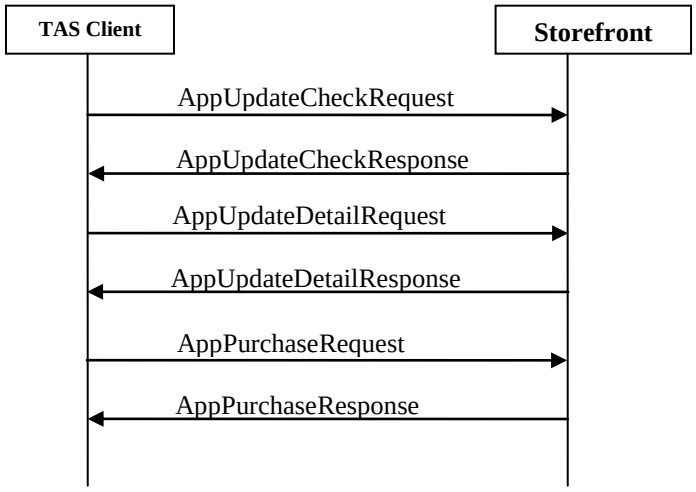

Figure 18: Application Update messages exchange

9.2.9.1 AppUpdateCheckRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
TerminalInfo	1	Structure	User's terminal information (Manufacturer,OS,Version)
AppUpdateCheck-List	1	Structure	Applications Version Update information

Table 38: AppUpdateCheckRequest message

Name	Cardinality	Data Type	Description
Manufacturer	1	String	The Manufacturer of The User's Terminal
TerminalType	1	String	The type of the terminal
Operate System	1	String	The Operate System of The User's Terminal
OS-Version	1	String	The OS-Version of The User's Terminal

Table 39: TerminalInfo structure

Name	Cardinality	Data Type	Description
App-Count	1	Integer	The Count of the Applications those will be checked Update
AppUpdateCheck-Item	0..n	Structure	The Application that will be checked Update

Table 40: AppUpdateCheck-List structure

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the Application that will be checked Update
App-Version	1	String	The Version of the Application that will be checked Update

Table 41: AppUpdateCheck-Item structure

9.2.9.2 AppUpdateCheckResponse

Name	Cardinality	Data Type	Description
Update-Available-Count	1	Integer	The Count of the Application that They have new Version
App-NewVer-infor-Package	0..N	Structure	The information packages of the Application that has new Version

Table 42: AppUpdateCheckResponse message

Name	Cardinality	Data Type	Description
App-ID	1	String	Application ID
App-New-Version	1	String	The New Version Information of the Application
Version-Date	1	Date	The Date of The day that the new version application start to be online

Table 43: App-NewVer-infor-Package Structure

9.2.9.3 AppUpdateDetailRequest

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the Application that will be checked Update
App-New-Version	1	String	The new version of the Application

Table 44: AppUpdateDetailRequest message

9.2.9.4 AppUpdateDetailResponse

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the Application
App-Logo	1	Binary	The LOGO Picture of the Application
App-Name	1	String	The Name of the Application
App-New-Version	1	String	The New Version Information of the Application
Version-Date	1	Date	The Date of The day that the new version application start to be online
Software-Size	1	String	The size of Install files
NewVer-Description	1	String	The Description of the NewVersion(s)
Price	1	Structure	The price of the update

Table 45: AppUpdateDetailResponse message

9.2.10 Application Sorting

This operation enables the TAS Client to perform application Sorting according to a criteria requested by TAS Client in the Storefront. It consists of two messages: An Application Sorting Request message from the TAS Client to the Storefront, and an Application Sorting Response message from the Storefront to the TAS Client.

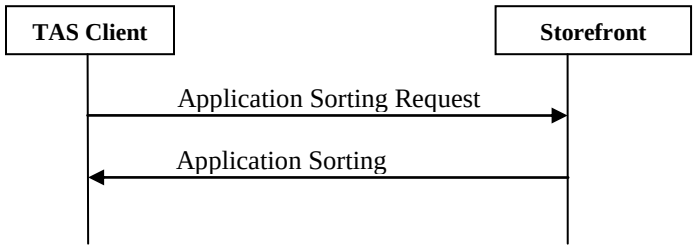

Figure 19: Application Sorting messages exchange

9.2.10.1 Application Sorting Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
Criteria	1	Integer	The criteria to apply on application sorting in the Storefront. A user can decide whether or not to use it and which criteria to apply on, such as, the latest version, the highest download rates within a specific time period, the price, etc.
Order	0..1	Boolean	0: ascent, 1: descent. If the parameter is missing, the default value is ascent.

Table 46: Application Sorting Request message

Value	Description
0	Name. It means the applications are sorted in the order of application name in alphabet order.
1	Price
2	Download number within a specific time period. Default time period is set by the TAS Service Provider.
3	Effective date
4	User's average rating score for the application.
5	Others. It should be readable and acceptable by the Storefront.

Table 47: Criteria Values

9.2.10.2 Application Sorting Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the Sorting Result; An error code will be provided when the sorting fails, such as incorrect criteria parameters.
AppInfoList	0..1	Structure	List of application descriptor(s) based on the applied criteria.

Table 48: Application Sorting Response message

9.2.11 Application Search (Optional)

This operation enables the TAS Client to perform application search on the Storefront.

It consists of two messages: An Application Search Request message from the TAS Client to the Storefront, and an Application Search Response message from the Storefront to the TAS Client.

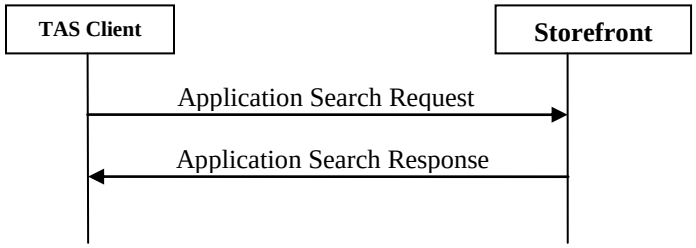

Figure 20: Application Search messages exchange

9.2.11.1 Application Search Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
QuickSearch	0..1	String	Related attributes applied on the application search in the Storefront. e.g. application name, application category. When Quick Search is applied, the information used for search is only a singlestring. QuickSearch and AdvancedSearch are mutually exclusive.
Advanced Search	0..1	Structure	Related attributes applied on the application search in the Storefront. e.g. application name, application category When Advanced Search is applied, the attribute used for search can be a combination of several attributes. QuickSearch and AdvancedSearch are mutually exclusive

Table 49: Application Search Request message

Name	Cardinality	Data Type	Description
MinPrice	0..1	Price	The minimum price
MaxPrice	0..1	Price	The maximum price
Developer	0..1	String	The name of the Developer
AppType	0..1	Structure	The type of the application

Table 50: AdvancedSearch structure

9.2.11.2 Application Search Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the Search Result; An error code will be provided when the Searching fails, such as incorrect attributes applied.
AppInfoList	0…1	Structure	List of application descriptor(s)

Table 51: Application Search Response message

9.2.12 User Deactive

This operation enables the User to deactivate in the Storefront. It consists of two messages: An UserDeactiveRequest message from the TAS Client to the Storefront, and an UserDeactiveResponse message from the Storefront to the TAS Client.

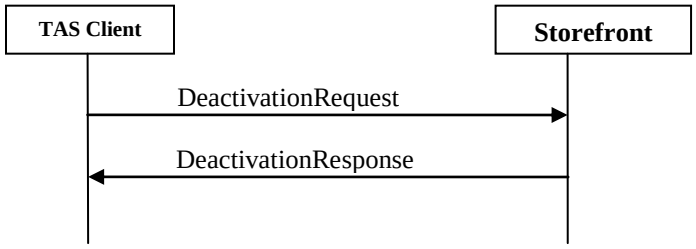

Figure 21: UserLogin messages exchange

9.2.12.1 DeactivationRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserPasswd	0..1	String	The Passwd of User

Table 52: UserDeactiveRequest message

9.2.12.2 DeactivationResponse

Name	Cardinality	Data Type	Description
result	1	structure	The Result status of the UserDeactivation

Table 53: UserDeactiveResponse message

9.2.13 Application Refund

This operation enables the User to Refund Application in the Storefront. It consists of two messages: An AppRefundRequest message from the TAS Client to the Storefront, and an AppRefundResponse message from the Storefront to the TAS Client.

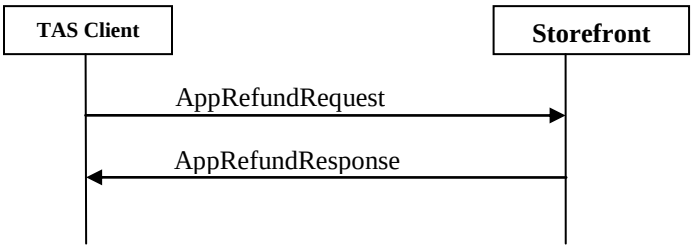

Figure 22: Application Refund messages exchange

9.2.13.1 AppRefundRequest

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserPasswd	0..1	String	The Passwd of User
RefundAppID	1	String	ID of Application to be refunded
Price	1	integer	The value of the Refunded applicaiton
RefundReason	1	Enumeration	The reason for the refunding (TBD)

Table 54: AppRefundRequest message

9.2.13.2 AppRefundResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of the AppRefund

Table 55: AppRefundResponse message

9.2.14 Application Download Status Report

This operation enables the TAS Client to report the status of the downloaded application to the Storefront.
It consists of two messages: An AppdownloadStatusReport message from the TAS Client to the Storefront, and an AppdownloadStatusResponse message from the Storefront to the TAS Client.

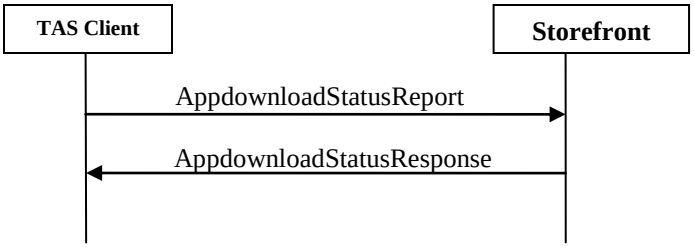

Figure 23: Application Download Status Report messages exchange

9.2.14.1 AppDownloadStatusReport

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS user
ApplicationID	1	String	The ID of the application
AppDownloadResultus	1	Structure	Application Download result

Table 56: AppdownloadStatusReport message

Name	Cardinality	Data Type	Description
AppDownloadStatus	1	Enumeration	The download status
Description	0..1	String	The detailed information about the status code

Table 57: AppDownloadResultus Structure

9.2.14.2 AppDownloadStatus Response

Name	Cardinality	Data Type	Description
Result	0..n	Result	Result of application download status

Table 58: AppDownloadStatusResponse message

9.2.15 Application Detail

This operation enables the User to request each application's details from the Storefront. It consists of two messages: An AppDetailRequest message from the TAS Client to the Storefront, and an AppDetailResponse message from the Storefront to the TAS Client.

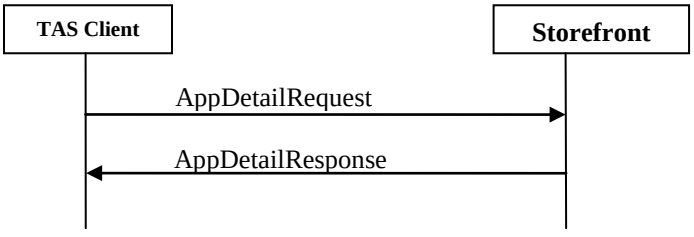

Figure 24: Application Detail messages exchange

9.2.15.1 AppDetailRequest

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the Application that will be reviewed

Table 59: AppDetailRequest message

9.2.15.2 AppDetailResponse

Name	Cardinality	Data Type	Description
AppInfo	1	Structure	The detailed Information of an application

Table 60: AppDetailResponse message

9.2.16 Application FeedbackList View (Optional)

This operation enables the TAS Client to get the application feedback list from the Storefront.

It consists of two messages: An Application FeedbackList Request message from the TAS Client to the Storefront, and an Application FeedbackList Response message from the Storefront to the TAS Client.

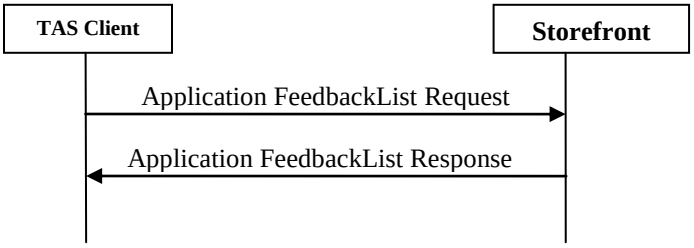

Figure 25: Application FeedbackList messages exchange

9.2.16.1 Application FeedbackList Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
ApplicationID	1	String	The ID of the Application

Table 61: Application FeedbackList Request message

9.2.16.2 Application FeedbackList Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The Status of the application feedackList
FeedbackCount	1	Integer	Number of Feedbacks for Application in this list
AverageRate	1	Integer	Derived from total average of user's rating (Total of User Rate / FeedbackCount)
FeedbackList	0..N	Structure	List of application feedbacks

Table 62: Application FeedbackList Response message

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the application
Version	0..1	String	The version of the application
UserID	1	String	The ID of the TAS User who made the comment
Rate	1	Integer	The Rate of the application feedback
Date	1	String	The date of the comment
Comment	1	String	The Comment content of the application feedback

Table 63: FeedbackList structure

9.2.17 Application Purchase Record

This operation enables the TAS Client to request for its purchase record within certain period in the Storefront.

It consists of two messages: An AppPurchaseRecordRequest message from the TAS Client to the Storefront, and an AppPurchaseRecord Response message from the Storefront to the TAS Client.

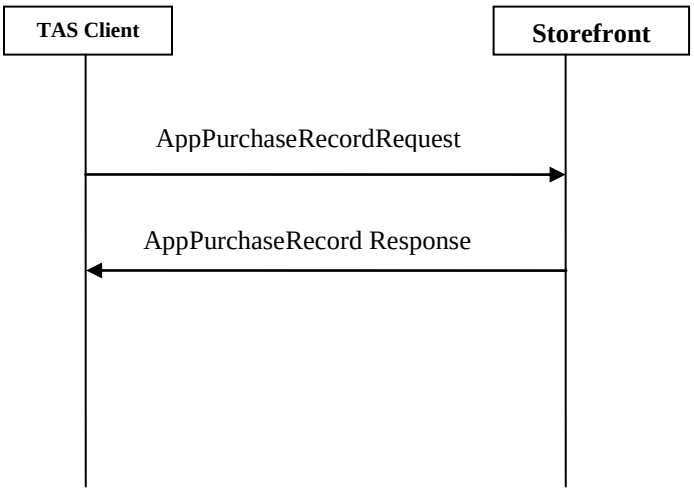

Figure 26: Application Purchase Record Messages Exchange

9.2.17.1 Application Purchase Record Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
StartDate	0..1	Date	Start date of the period
EndDate	0..1	Date	End date of the period

Table 64: Application Purchase Record Request message

9.2.17.2 Application Purchase Record Response

Name	Cardinality	Data Type	Description
StoreID	1	String	The ID of the Storefront
AppPurchase InfoList	0…1	Structure	List of application(s) in the Purchase information

Table 65: Application Purchase Record Response message

Name	Cardinality	Data Type	Description
AppCount	1	Integer	Number of applications in this list
AppPurchaseInfo	0..N	Structure	The application Purchase information

Table 66: AppPurchaseInfoList structure

Name	Cardinality	Data Type	Description
Application-ID	1	String	The ID of the application
Name	1	String	The name of the application
PurchaseDate	1	Date	Purchase date of the period
Price	1	Structure	Price of the Application to Users

Table 67: AppPurchaseInfo structure

9.2.18 Application Favorites List Creation

This operation enables the TAS Clients to create its own Favorites application lists.

It consists of two messages: An AppFavoritesListCreation Request message from the TAS Client to the Storefront, and an AppFavoritesListCreation Response message from the Storefront to the TAS Client.

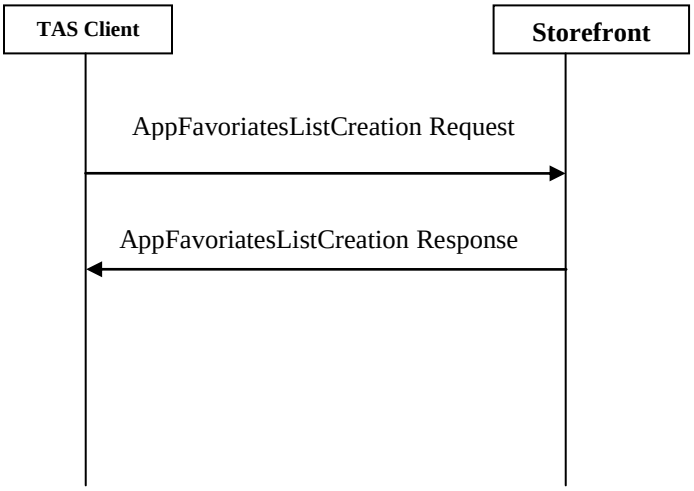

Figure 27: Application Favourites List Creation Messages Exchage

9.2.18.1 Application Favourites List Creation Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
AppFavoriates List	1	Structure	List of the application favorites be created.

Table 68: Application Favorites List Creation Request message

Name	Cardinality	Data Type	Description
AppIDCount	1	Integer	Number of application IDs in this list
ApplicationID	1..N	String	The ID of the application

Table 69: AppFavorites ID List structure

9.2.18.2 Application Favoriates List Creation Response

Name	Cardinality	Data Type	Description
Result	1	Structure	Status code and error information

Table 70: Application Favoriates List Creation Response message

9.2.19 Application Favorites List Deletion

This operation enables the TAS Clients to delete its own favorites application list created.

It consists of two messages: An AppFavoriatesListDeletoin Request message from the TAS Client to the Storefront, and an AppFavoriatesListDeletoin Response message from the Storefront to the TAS Client.

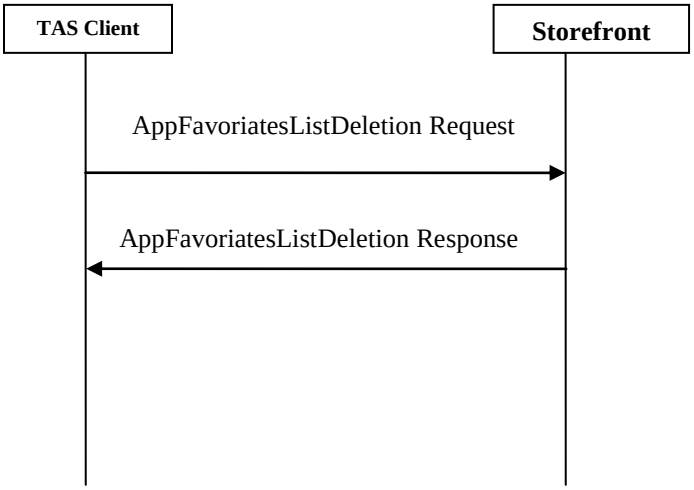

Figure 28: Application Faroriates List Deletion Messages Exchange

9.2.19.1 Application Favoriates List Deletion Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
AppFavorites List	1	Structure	List of the application favoriates be deleted

Table 71: Application Favorites List Deletion Request message

9.2.19.2 Application Favorites List Deletion Response

Name	Cardinality	Data Type	Description
Result	1	Structure	Status code and error information

Table 72: Application Favorites List Deletion Response message

9.2.20 Application Favorites List Update

This operation enables the TAS Clients to update its own Favorites application lists.

It consists of two messages: An AppFavoriatesListUpdate Request message from the TAS Client to the

Storefront, and an AppFavoriatesListUpdate Response message from the Storefront to the TAS Client.

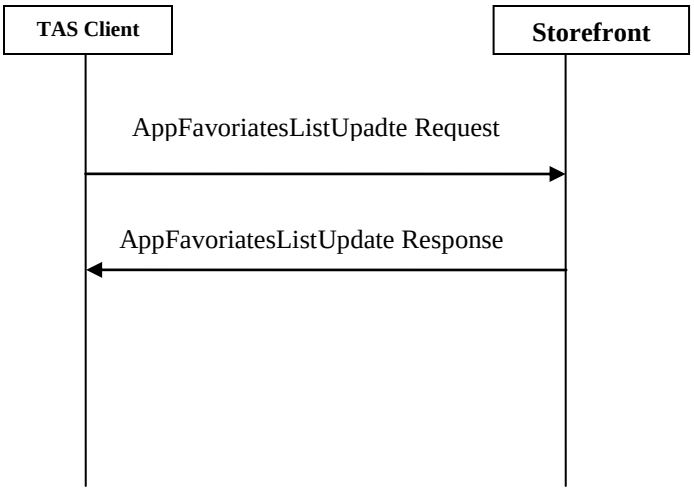

Figure 29: Application Favorites List Update Messages Exchange

9.2.20.1 Application Favorites List Update Request

Name	Cardinality	Data Type	Description
ClientID	1	String	The ID of the TAS Client
AppFavoriates Update List	1	Structure	List of the application favoriates be updated

Table 73: Application Favoriates List Deletion Request message

Name	Cardinality	Data Type	Description
AppIDCount	1	Integer	Number of application IDs in this list
ApplicationID	1..N	String	The ID of the application
UpdateAction	1	Enumeration	Action to apply on this application: Add or Remove

Table 74: Application Favoriates Update ID List Structure

Enumeration	Description
Add	The application will be added to the favoriates list.
Remove	The application will be removed from the favoriates list.

Table 75: UpdateAction Enumeration

9.2.20.2 Application Favoriates List Update Response

Name	Cardinality	Data Type	Description
Result	1	Structure	Status code and error information

Table 76: Application Favoriates List Update Response message

9.2.21 Application Gift (Optional)

This operation enables the TAS Clients to send application gift to the receiveuser.

It consists of two messages: An AppGiftRequest message from the TAS Client to the Storefront, and an AppGiftResponse message from the Storefront to the TAS Client.

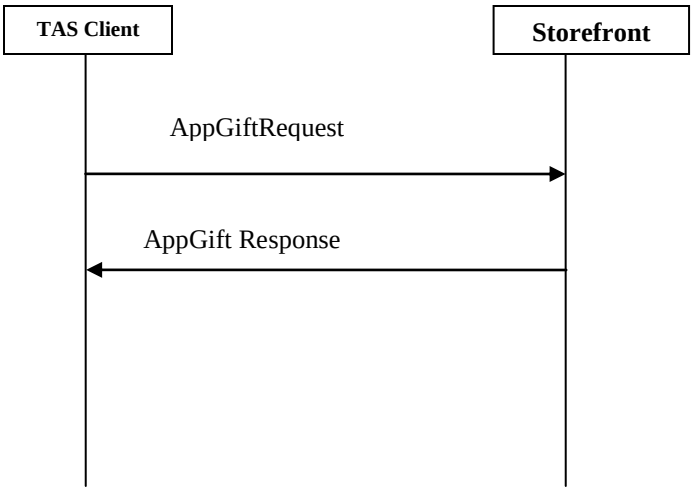

Figure 30: Application Gift Messages Exchange

9.2.21.1 Application Gift Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the User
Application-ID	1	String	The ID of the gift application
ReceiveUserID	1	String	The ID of the gift Receive User
Giftmessage	1	String	The content of the gift message
Date	1	Date	The date of gift

Table 77: Application Gift Request message

9.2.21.2 Application Gift Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the application gift

Table 78: Application gift Response message

9.2.22 Application Recommendation

This operation enables the TAS Clients to send application Recommendation to the receiveuser.

It consists of two messages: An AppRecomRequest message from the TAS Client to the Storefront, and an AppRecomResponse message from the Storefront to the TAS Client.

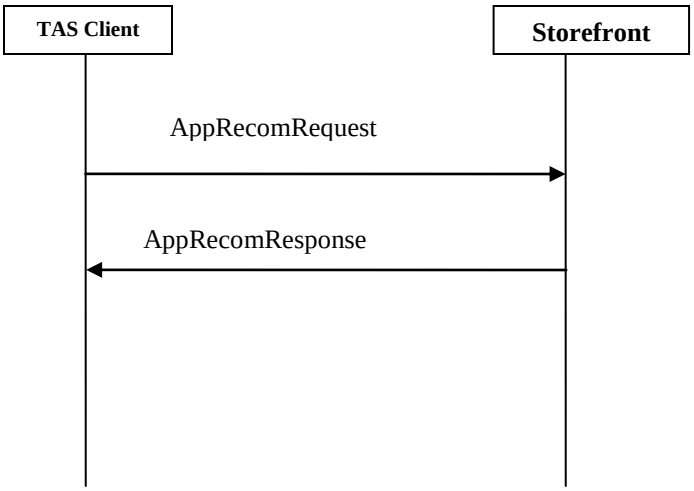

Figure 31: Application Recommentdation Messages Exchange

9.2.22.1 Application Recommendation Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the User
Application-ID	1	String	The ID of the Recommendation application
ReceiveUserID	1	String	The ID of the Recommendation Receive User
Recommessage	1	String	The content of the Recommendation message
Date	1	Date	The date of Recommendation

Table 79: AppRecomRequest message

9.2.22.2 Application Recommendation Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the application Recommendation

Table 80: AppRecomResponse message

9.2.23 Gift Reception (Optional)

This operation enables the TAS Clients to request for its Giftlist in the Storefront.

It consists of two messages: An GiftlistRequest message from the TAS Client to the Storefront, and an GiftlistResponse message from the Storefront to the TAS Client.

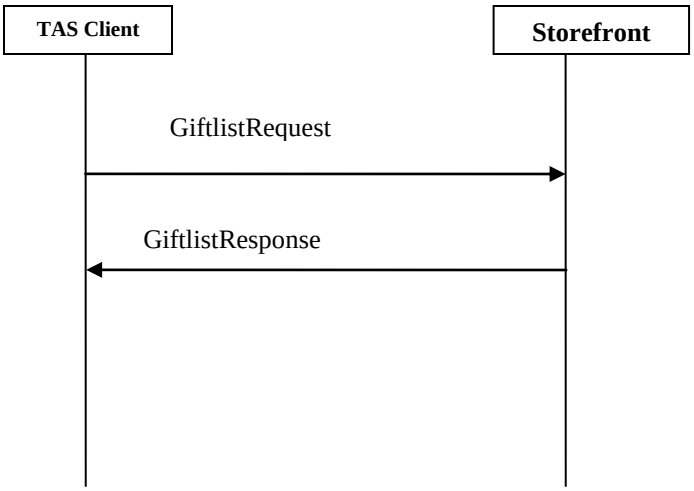

Figure 32: Gift Reception Messages Exchange

9.2.23.1 Giftlist Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the User

Table 81: Giftlist Request message

9.2.23.2 Giftlist Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the Giftlist Request result
Giftlist InfoList	0..1	Structure	List of application(s) in the Giftlist

Table 82: Giftlist Response message

Name	Cardinality	Data Type	Description
AppCount	1	Integer	Number of applications in this list
GiftlistInfo	0..N	Structure	The application Giftlist information

Table 83: Giftlist Info structure

Name	Cardinality	Data Type	Description
Application-ID	1	String	The ID of the application
Name	1	String	The name of the application
GiftUserID	1	String	The ID of the GiftUser
Price	1	Structure	Price of the Application to Users
Version	1	String	A version number typically in the format of "X.Y", in which X and Y are integers.
AppInfo	0..1	Structure	The detailed Information of an application

Table 84: GiftlistInfo structure

9.2.24 Application Shopping Cart List Creation (Optional)

This operation enables the TAS Client to create Application Shopping Cart Lists on the Storefront.

It consists of two messages: An AppShoppingCartListCreation Request message from the TAS Client to the Storefront, and an AppShoppingCartListCreation Response message from the Storefront to the TAS Client.

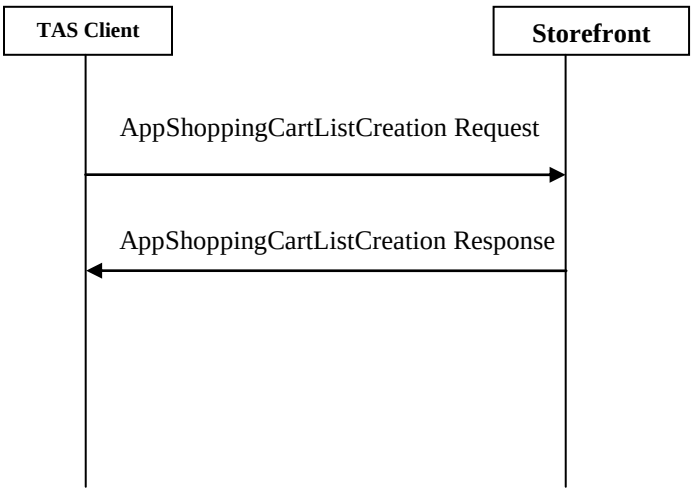

Figure 33: AppShoppingCartListCreation Messages Exchange

9.2.24.1 AppShoppingCartListCreation Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
AppIDList	1	Structure	List of application ID(s) put in the shopping cart

Table 85: AppShoppingCartListCreation Request message

9.2.24.2 AppShoppingCartListCreation Response

Name	Cardinality	Data Type	Description
Result	1	Structure	Status code and error information

Table 86: AppShoppingCartListCreation Response message

9.2.25 Application Shopping Cart List Modification (Optional)

This operation enables the TAS Client to modify the Application Shopping Cart List on the Storefront.

It consists of two messages: An AppShoppingCartListModification Request message from the TAS Client to the Storefront, and an AppShoppingCartListModification Response message from the Storefront to the TAS Client.

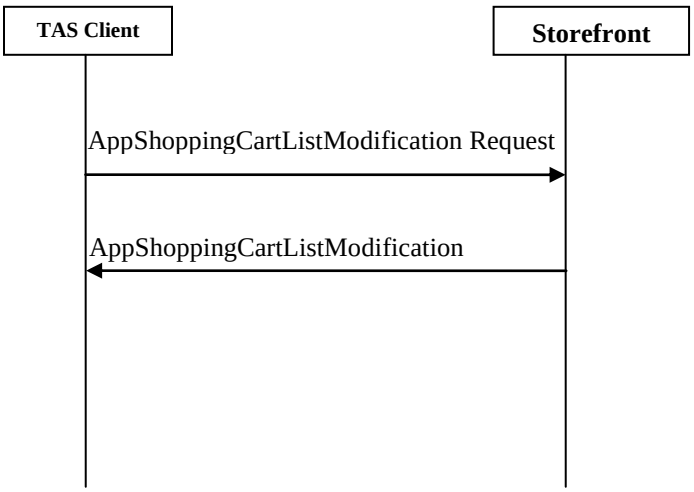

Figure 34: AppShoppingCartListModification Messages Exchange

9.2.25.1 AppShoppingCartListModification Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
AppShopCartModifyList	1	Structure	List of application (s) the user wants to modify in the shopping cart

Table 87: AppShoppingCartListModification Request message

Name	Cardinality	Data Type	Description
opFlag	1	Integer	1:add 2:delete
AppIDList	1	Structure	List of application ID(s) the user wants to add/delete in the shopping cart

Table 88: AppShopCartModifyList structure

9.2.25.2 AppShoppingCartListModification Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of Application Shopping Cart List Modification

Table 89: AppShoppingCartListModification Response message

9.2.26 IAP Item List (Optional)

This operation enables the TAS Client to get IAP item list on the Storefront.

It consists of two messages: An IAPItemListRequest message from the TAS Client to the Storefront, and an IAPItemListResponse message from the Storefront to the TAS Client.

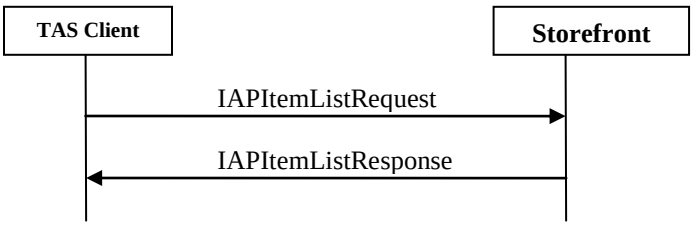

Figure 35: IAP Item List messages exchange

9.2.26.1 IAP Item List Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
ApplicationID	1	String	The ID of the Application that will be checked IAP items

Table 90: IAP Item List Request message

9.2.26.2 IAP Item List Response

Name	Cardinality	Data Type	Description
IAPItemCount	1	Integer	Number of the IAP Items for Application in this list
IAPInfo	0..N	Structure	The detailed Information of an application's IAP items

Table 91: IAP Item List Response message

9.2.27 IAP Item Purchase (Optional)

This operation enables the TAS Client to purchase IAP item in the Storefront.

It consists of two messages: An IAPItemPurchaseRequest message from the TAS Client to the Storefront, and an IAPItemPurchaseResponse message from the Storefront to the TAS Client.

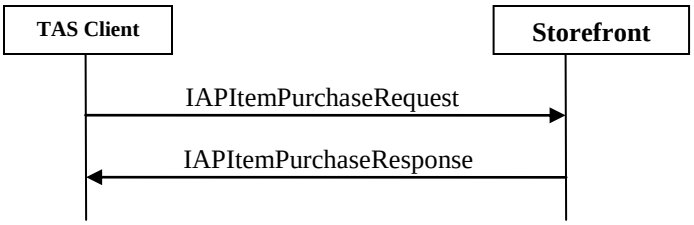

Figure 36: IAP Item Purchase messages exchange

9.2.27.1 IAP Item Purchase Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserPasswd	0..1	String	The Passwd of User
IAPItemIDList	1	Structure	List of IAP Items to be purchased

Table 92: IAP Item Purchase Request message

Name	Cardinality	Data Type	Description
IAPItemIDCount	1	Integer	Number of IAP item IDs in this list
IAPItemID	0..N	String	The ID of the IAP item

Table 93: IAPItemIDList structure

9.2.27.2 IAP Item Purchase Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of the IAP item Purchase

Table 94: IAP Item Purchase Response message

9.2.28 Application Shopping Cart List Inquiry (Optional)

This operation enables the TAS Client to obtain the Application Shopping Cart List from the Storefront.

It consists of two messages: An AppShoppingCartListInquiry Request message from the TAS Client to the Storefront, and an AppShoppingCartListInquiry Response message from the Storefront to the TAS Client.

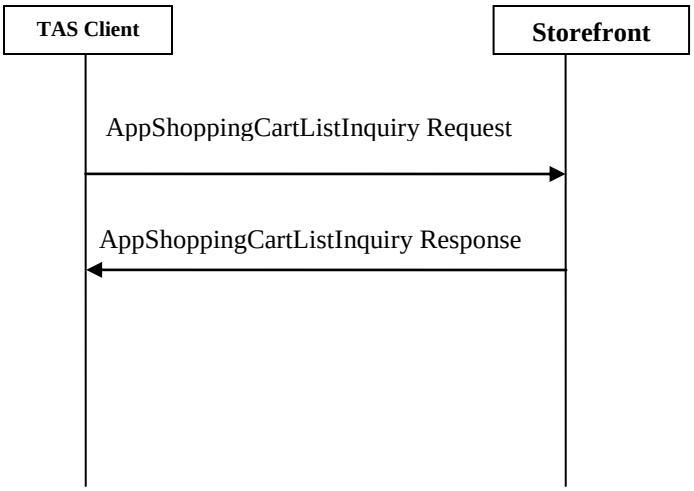

Figure 37: AppShoppingCartListInquiry Messages Exchange

9.2.28.1 AppShoppingCartListInquiry Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
UserPasswd	0..1	String	The Password of User

Table 95: AppShoppingCartListInquiry Request message

9.2.28.2 AppShoppingCartListInquiry Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of Application Shopping Cart List Inquiry
AppInfoList	1	Structure	List of application (s) put in the shopping cart Condition: This parameter is only with the Value of the Status of Result parameter is Success.

Table 96: AppShoppingCartListInquiry Response message

9.2.29 Application Begging (Optional)

This operation enables the TAS Client to send Application Begging Request to the Storefront, to beg the application from a friend or a parent.

It consists of two messages: An Application Begging Request message from the TAS Client to the Storefront, and an Application Begging Response message from the Storefront to the TAS Client.

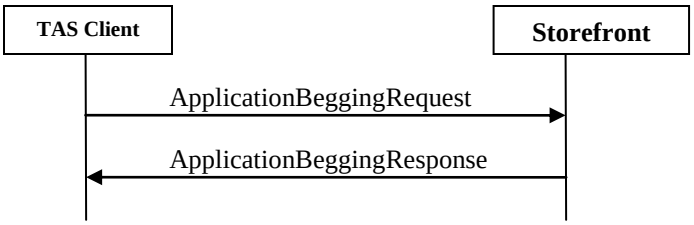

Figure 38: Application Begging messages exchange

9.2.29.1 Application Begging Request

Name	Cardinality	Data Type	Description
UserID	1	String	The ID of the TAS User
ApplicationID	1	String	The ID of the application to be beg
Giver-ID	1	String	The ID of the Giver
BeggingMessage	1	String	The content of the begging message
Date	1	Date	The date of begging

Table 97: Application Begging Request message

9.2.29.2 Application Begging Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of the Application Begging

Table 98: Application Begging Response message

9.3 TAS-3

This interface is postponed to the future release.

9.4 TAS-4

This interface is postponed to the future release.

9.5 TAS-5

9.5.1 Application Deletion

This operation enables the TAS Developers to delete its application on the Storefront.

It consists of two messages: Application Deletion Request and Application Deletion Response. An Application Deletion Request message from the Developer to the Developer Support, this message will then be delivered to the Storefront, which can be found in 8.1 TAS-1 as well.

Application Deletion Response is sent from the Storefront to the Developer Support, which can be found in 8.1 TAS-1 as well. This Application Deletion Response message will then be delivered from the Developer Support to the Developers.

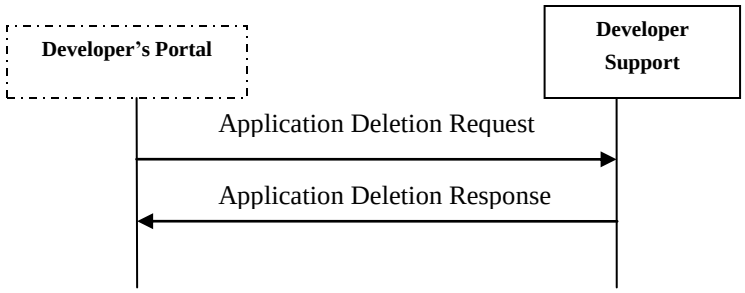

Figure 39: Application Deletion messages exchange

9.5.1.1 Application Deletion Request

Name	Cardinality	Data Type	Description
Developer-ID	1	String	The ID of the TAS Developer.
AppID	1	String	Application ID to be deleted
Reason	0..1	String	reason to delete this application

Table 99: Application Deletion Request message

9.5.1.2 Application Deletion Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the deletion Result.

Table 100: Application Deletion Response message

9.5.2 Application Status Check

This operation enables the TAS Developer to interact with the Developer Support to check the audit status of the the applications and IAP Items developed by itself.
It consists of two messages: an Application Status Check Request from the developer's Portal to the Developer Support, and an Application Status Check Response from the Developer Support to the Developer's Portal.

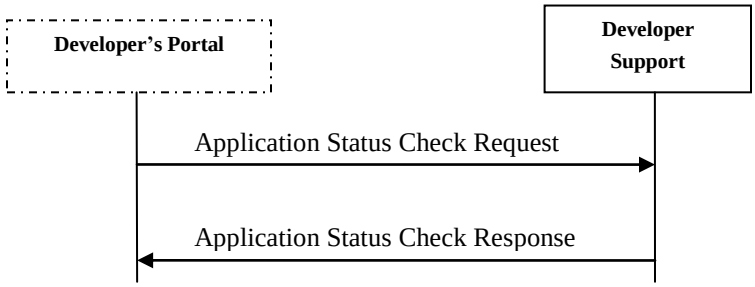

Figure 40: Application Status Check Messages exchange

9.5.2.1 Application Status Check Request

Name	Cardinality	Data Type	Description
Developer-ID	1	String	The ID of the TAS Developer.
AppID	1	String	Application ID to be checked

Table 101: Application Status Check Request message

9.5.2.2 Application Status Check Response

Name	Cardinality	Data Type	Description
AppStatus	1	Enumeration	The application audit status
Reason	0..1	String	Reason for the application not being able to be to be on online

Table 102: Application Deletion Response message

9.5.3 Application Upload

This operation enables the TAS Developers to upload its application on the Developer Support.

It consists of two messages: Application Upload Request and Application Upload Response. An Application Upload Request message is sent from the Developer to the Developer Support. And the Application Upload Response message is sent from the Developer Support to the developer.

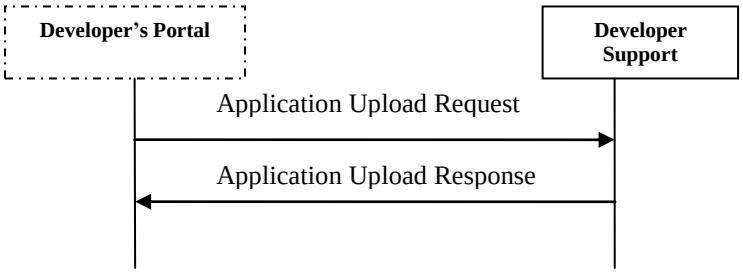

Figure 41: Application Upload messages exchange

9.5.3.1 Application Upload Request

Name	Cardinality	Data Type	Description
ApplicationName	1	String	Application Name
DeveloperId	1	String	ID of the Developer
AppType	1	Structure	Type of the Application, can be "games", "contents", "tools", "entertainment", "books", etc.
Description	1	String	Some text description to illustrate the Application
Version	1	String	A version number typically in the format of "X.Y", in which X and Y are integers.
SubmitTime	1	Time	The time at which this Application was submitted to the Developer Support
EffectiveDate	1	Date	The date on which the Application begins to be available to Users
ExpireDate	1	Date	The date on which the Application will expire
Price	1	Structure	Price of the Application to Users
Logo	0…1	Binary	The logo picture of the Application
Language	1	Enumerate	The Language of the Application. It should use to "ISO-639"Language code standard. (e.g. en, ko, zh, jp, all) http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
Size	1	Integer	The size of Application Install files in bytes.
Whatsnew	1	String	The description of what's new in latest version
Screenshoot	1	Structure	The screenshots of the Application

Table 103: Application Upload Request message

9.5.3.2 Application Upload Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the deletion Result.
ApplicationID	0	String	If the result is success, the Application ID will be generalized by the Developer Support and sent to the developer in the Application Upload Response message.

Table 104: Application Upload Response message

9.5.4 IAP item Add request

This operation enables the TAS Developers to add IAP items of his application. The IAP items also have the state just like the application. And it need to be checked by Developer Support. For exam0ple, after Developer Support check it successfully, its status becomes online, the corresponding information will be synchronized to StoreFront so that the end user could purchase.

It consists of two messages: IAP item add Request and IAP item add Response. An IAP item add Request message from the Developer to the Developer Support IAP item add Response is sent from the Developer Support to the developer

9.5.4.1 IAP item Add Request

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the corresponding Application
IAPitemName	1	String	The name of the IAP item
DeveloperId	1	String	ID of the Developer
Description	1	String	Some text description to illustrate the Application
SubmitTime	1	Time	The time at which this Application was submitted to the Developer Support
EffectiveDate	1	Date	The date on which the IAPitem begins to be available to Users
ExpireDate	1	Date	The date on which the IAPitem will expire
Needdownload	1	Boolean	1 : the IAP item need download new content 0: no need to download new content
DownloadURL	0	String	If needdownload is “1”, DownloadURL should be indicated.
Price	1	Structure	Price of the Application to Users

Table 105: IAP Item Add Request message

9.5.4.2 IAP item Add Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the IAP Item Add Result.
IAPitemID	0	String	If the result is success, the IAPitem ID will be generalized by the Developer Support and sent to the developer in the IAP Item Add Response message.

Table 106: IAP Item Add Response message

9.5.5 IAP Item Modify request

This operation enables the TAS Developers to modify IAP items of his application. It consists of two messages: IAP modify Request and IAP modify Response. An IAP modify Request message from the Developer to the Developer Support IAP modifyResponse is sent from the Developer Support to the developer

9.5.5.1 IAP Item Modify Request

Name	Cardinality	Data Type	Description
IAPitemName	0..1	String	The name of the IAP item
Description	0..1	String	Some text description to illustrate the Application
SubmitTime	0..1	Time	The time at which this Application was submitted to the Developer Support
EffectiveDate	0..1	Date	The date on which the IAPitem begins to be available to Users
ExpireDate	0..1	Date	The date on which the IAPitem will expire
Price	0..1	Structure	Price of the Application to Users
Needdownload	0..1	Boolean	: the IAP item need download new content 0: no need to download new content
DownloadURL	0..1	String	If needdownload is "1", DownloadURL should be indicated.

Table 107: IAP Item Modify Request message

9.5.5.2 IAP Item Modify Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the IAP Modify Result.

Table 108: IAP Item Modify Response message

9.5.6 IAP Item Delete request

This operation enables the TAS Developers to delete IAP items of his application. It consists of two messages: IAP delete Request and IAP delete Response.

9.5.6.1 IAP Item Delete Request

Name	Cardinality	Data Type	Description
IAPitemID	1	String	The ID of the IAP item

Table 109: IAP Item Delete Request message

9.5.6.2 IAP delete Response

Name	Cardinality	Data Type	Description
Result	1	Structure	The status of the IAP delete Result.

Table 110: IAP Item Delete Response message

9.5.7 Application Sales Report

This operation enables the TAS Developer to interact with the Developer Support to get the sales report of the applications and IAP Items developed by itself.

It consists of two messages: an Application Sales Report Request from the developer's Portal to the Developer Support, and an Application Sales Report Response from the Developer Support to the Developer's Portal.

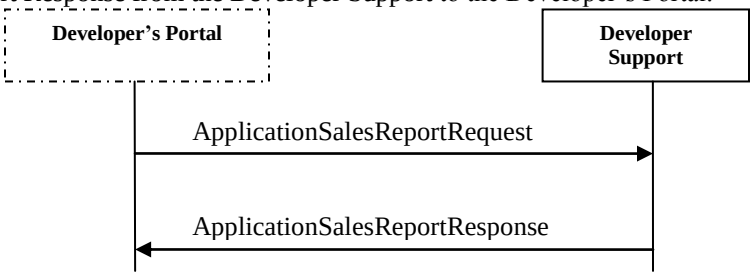

Figure 42: Application Sales Report messages exchange

9.5.7.1 ApplicationSalesReportRequest

Name	Cardinality	Data Type	Description
DeveloperID	1	String	The ID of the TAS Developer.
StartDate	0..1	Date	Start date of the period
EndDate	0..1	Date	End date of the period

Table 111: ApplicationSalesReportRequest message

9.5.7.2 ApplicationSalesReportResponse

Name	Cardinality	Data Type	Description
AppCount	1	Integer	Number of applications in this list
AppSalesReport	0..N	Structure	The application Sales Report

Table 112: ApplicationSalesReportResponse message

9.5.8 Developer Registration

This operation enables the TAS Developers to register the developer profile on the Developer Support.
It consists of two messages: a Developer Registration Request from the developer's Portal to the Developer Support, and a Developer Registration Response from the Developer Support to the Developer's Portal.

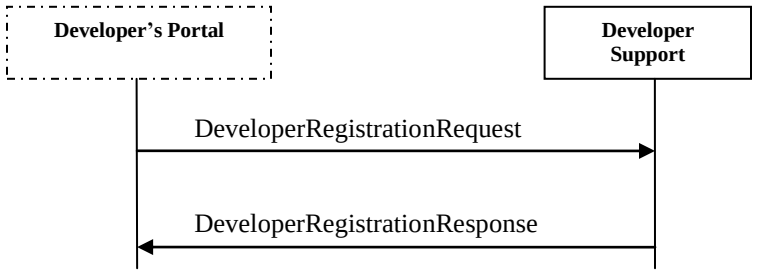

Figure 43: Application Sales Report messages exchange

9.5.8.1 DeveloperRegistrationRequest

Name	Cardinality	Data Type	Description
DeveloperID	1	String	ID of the Developer
DeveloperType	1	Enumerate	The type of developer. It can be "Individual", "group", "Enterprise", etc.
PortalUserID	1	String	User ID of Developer Portal
PortalUserPasswd	1	String	User Password of Developer Portal
FirstName	0..1	String	First Name of Developer
LastName	0..1	String	Last Name of Developer
Email	1	String	Email Address of Developer
CompanyHomepage	0..1	String	Web Site URL of Developer
CompanyName	1	String	It is the name that will show up as the name of the developer of your applications
PostalAddress	0..1	String	Postal Address of Developer
PostalCode	0..1	String	Postal Code of Address
Country	1	Enumerate	The country of developer. It should use to [ISO-3166 Alpha-2] regional code standard. (e.g. US, KR, CN, JP, ALL).
PhoneNumber	0..1	String	Phone Number of Developer
ContractInfo	0..N	Structure	Contract Information

Table 113: DeveloperRegistrationRequest message

9.5.8.2 DeveloperRegistrationResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of the Developer Registration

Table 114: DeveloperRegistrationResponse message

9.5.9 Developer Deregistration

This operation enables the Developer to deregister in the Developer Support.
It consists of two messages: a DeveloperDeregistrationRequest from the developer's Portal to the Developer Support, and a DeveloperDeregistrationResponse from the Developer Support to the Developer's Portal.

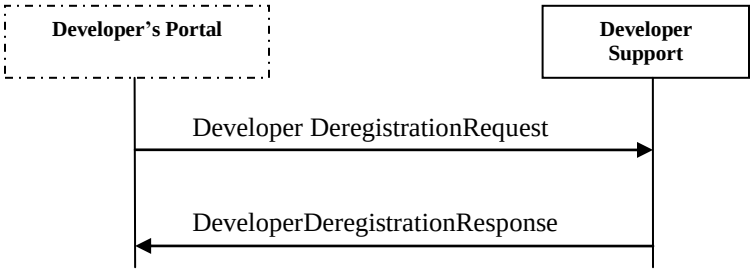

Figure 44: Developer Deregistration messages exchange

9.5.9.1 DeveloperDeregistrationRequest

Name	Cardinality	Data Type	Description
PortalUserID	1	String	User ID of Developer Portal
PortalUserPassw d	1	String	User Password of Developer Portal
DeveloperID	1	String	ID of the Developer

Table 115: DeveloperDeregistrationRequest message

9.5.9.2 DeveloperDeregistrationResponse

Name	Cardinality	Data Type	Description
Result	1	Structure	The Result status of the DeveloperDeregistrationRequest

Table 116: DeveloperDeregistrationResponse message

9.6 TAS-6

9.6.1 Malicious Application Notification

This operation enables the storefront to send Malicious Application Notification to the Developer Support, to provide information on Malicious applications which have been reported.

It consists of two messages: A Malicious Application Notification message from the Storefront to the Developer Support, and a Malicious Application Notification Response message from the Developer Support to the Storefront.

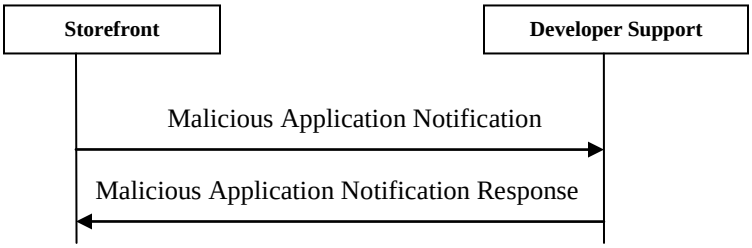

Figure 45: Malicious Application Notification Messages Exchange

9.6.1.1 Malicious Application Notification

Name	Cardinality	Data Type	Description
StoreID	1	String	The ID of the Store Front
CommentList	1	Structure	List of application ID(s) to be notified.

Table 117: Malicious Application Notification message

Name	Cardinality	Data Type	Description
AppIDCount	1	Integer	Number of application IDs in this list
ApplicationID	0..N	String	The ID of the application
Comment	0..1	String	Reason to notify this Malicious Application, or suggestion to handle this Malicious Application

Table 118: CommentList structure

9.6.1.2 Malicious Application Notification Response

Name	Cardinality	Data Type	Description
result	1	Structure	Status code and error information

Table 119: Malicious Application Notification Response message

9.6.2 Application Sale Information

This operation enables the Storefront to send Application Sale Information Notification Message to the Developer Support to provide sale information of the applications provide by the developer support, such as how many times of download with certain period, how many times of complains with certain period, how many times of application refund with certain period, average rate within certain period, and comments received.

It consists of two messages: An Application Sale Information Notification Message from the Storefront to the Developer Support, and An Application Sale Information Response from the Developer Support to the Storefront.

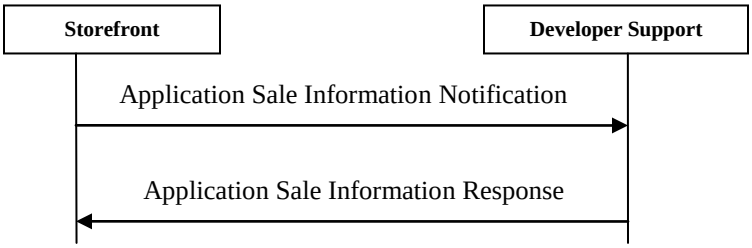

Figure 46: Application Sale Information message exchange

9.6.2.1 Application Sale Information Notification

Name	Cardinality	Data Type	Description
StoreID	1	String	The ID of the Storefront
AppSaleInfoList	0..1	Structure	List of application(s) in the sale information

Table 120: Application Sale Information Notification message

Name	Cardinality	Data Type	Description
AppCount	1	Integer	Number of applications in this list
AppSaleInfo	0..N	Structure	The application sale information

Table 121: AppSaleInfoList structure

Name	Cardinality	Data Type	Description
Application-ID	1	String	The ID of the application
Name	1	String	The name of the application
StartDate	1	Date	Start date of the period
EndDate	1	Date	End date of the period
downloadInfo	1	Integer	How many times of downloads within certain period
compalinInfo	1	Integer	How many times of complains within certain period
RefundInfo	1	Integer	How many times of refund within certain period
RateInfo	1	Integer	The average rate given within certain period
Comment	0..1	String	The Comment content of the application received

Table 122: AppSaleInfo structure

9.6.2.2 Application Sale Information Response

Name	Cardinality	Data Type	Description
Result	1	Structure	Status code and error information

Table 123: Application Sale Information Response Message

9.6.3 Application State Transition Notification

This operation enables the Storefront to Notify the Developer Support that the state of the application has been shifted from one state to the other state.

It consists of two messages: An Application state transition Notification message from the Storefront to the Developer Support, and an Application state transition Response message from the Developer Support to the Storefront.

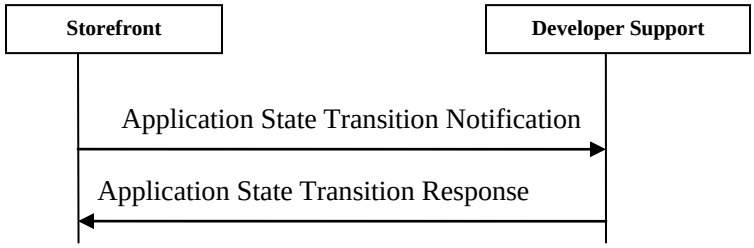

Figure 47: Application state transition Notification messages exchange

9.6.3.1 Application state transition Notification

Name	Cardinality	Data Type	Description
StoreID	1	String	The ID of the Store Front
AppCount	1	Integer	Number of Application which encounter state transition in this list
StatetransAppList	0..N	Structure	List of application ID(s) to be notified.

Table 124: Application state transition Notification message

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the application which encounter state transition
State	1	Enumeration	The description of the state transition
Reason	0..1	String	Reason of the state transition

Table 125: StatetransAppList structure

Enumeration	Description
online	The state is changed to “online” state
invalid	The state is changed to “invalidated” state

Table 126: State Enumeration

9.6.3.2 Application state transition Response

Name	Cardinality	Data Type	Description
result	1	Structure	Status code and error information

Table 127: Application state transition Response message

9.7 Common Structures

This section provide the common structures that be used by all the interfaces.

9.7.1 Result Structure

Name	Cardinality	Data Type	Description
statusCode	1	Enumeration	Result of user active request: 0: fail; 1: success
statusInfo	0..1	String	The detailed information about the error.

Table 128: Result Structure

9.7.2 AppInfoList Structure

Name	Cardinality	Data Type	Description
Count	1	Integer	Number of application IDs in this list
AppInfo	0..N	Stucture	The Information of an application

Table 129: AppInfoList structure

9.7.3 AppInfo Structure

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the Application
ApplicationName	1	String	Application Name
DeveloperId	1	String	ID of the Developer
Type	1	Enumerate	Type of the Application, can be “games”, “contents”, “tools”, “entertainment”, “books”, etc.
Description	1	String	Some text description to illustrate the Application
Status	1	Enumerate	Status can be “submitted”, “audited”, “tested”, “online”, “offline”, “end”, etc.
Version	1	String	A version number typically in the format of “X.Y”, in which X and Y are integers.
SubmitTime	1	Time	The time at which this Application was submitted to the Developer Support
EffectiveDate	1	Date	The date on which the Application begins to be available to Users
ExpireDate	1	Date	The date on which the Application will expire
Price	1	Structure	Price of the Application to Users
Logo	0..1	Binary	The logo picture of the Application
Language	1	Enumerate	The Language of the Application. It should use to “ISO-639” Language code standard. (e.g. en, ko, zh, jp, all) http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
Size	1	Integer	The size of Application Install files in bytes.
Whatsnew	1	String	The description of what’s new in latest version
Screenshoot	1	Structure	The screenshots of the Application

Table 130: AppInfo structure

Name	Cardinality	Data Type	Description
Value	1	Float	The amount of the price
Price-Unit-Code	1	Enumerate	1:USD;2:EUR;3:RMB ……

Table 131: Price structure

Name	Cardinality	Data Type	Description
ImageCount	1	Integer	Number of Screenshot image
ScreenshotImage	1..N	Image	Screenshot Image file

Table 132: Screenshoot structure

9.7.4 IAPInfo Structure

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the Application
IAPitemID	1	String	The ID of the IAP item
IAPitemName	1	String	The name of the IAP item
DeveloperId	1	String	ID of the Developer
Description	1	String	Some text description to illustrate the Application
Status	1	Enumerate	Status can be “submitted”, “audited”, “tested”, “online”, “offline”, “end”, etc.
SubmitTime	1	Time	The time at which this Application was submitted to the Developer Support
EffectiveDate	1	Date	The date on which the IAPitem begins to be available to Users
ExpireDate	1	Date	The date on which the IAPitem will expire
Price	1	Structure	Price of the Application to Users
Needdownload	1	Boolean	: the IAP item need download new content 0: no need to download new content
DownloadURL	1	String	If needdownload is “1”, DownloadURL should be indicated.

Table 133: IAPInfo structure

9.7.5 AppTypeList Structure

Name	Cardinality	Data Type	Description
count	1	String	The count of the apptype
AppType	0..n	Structure	The apptype of the Developer Support

Table 134: AppTypeList structure

9.7.6 AppType Structure

Name	Cardinality	Data Type	Description
AppTypeID	1	String	The ID of the application type
AppTypeName	1	String	The name of the application type
AppTypeDes	1	String	The description of the application type

Table 135: AppType structure

9.7.7 AppIDList Structure

Name	Cardinality	Data Type	Description
AppIDCount	0..1	Integer	Number of application IDs in this list
ApplicationID	0..N	String	The ID of the application

Table 136: AppIDList structure

9.7.8 TerminalInfo Structure

Name	Cardinality	Data Type	Description
TerminalBrand	1	String	The brand name of the terminal
TerminalType	1	String	The type of the terminal
OSType	0..1	String	The type of the operating system
OSVersion	0..1	String	The version of the operating system

Table 137: TerminalInfo Structure

9.7.9 AppSalesReport Strcture

Name	Cardinality	Data Type	Description
ApplicationID	1	String	The ID of the application
Name	1	String	The name of the application
Version	1	String	A version number typically in the format of "X.Y", in which X and Y are integers.
UnitSales	1..N	Structure	Total unit sales of the Application in each storefront
FeedbackCount	1	Integer	Number of Feedbacks for Application
AverageRate	1	Integer	Derived from total average of user's rating (Total of User Rate / FeedbackCount)
FeedbackList	0..N	Structure	List of application feedbacks

Table 138: AppSalesReport Structure

9.7.10 UnitSales Structure

Name	Cardinality	Data Type	Description
StoreID	1	String	The ID of the Storefront
SalesCount	1	Integer	Total Count of Sales
Price	1	Structure	Price of the Application to Users

Table 139: UnitSales structure

9.7.11 ContractInfo Structure

Name	Cardinality	Data Type	Description
ContractRegion	1	Enumerate	The Contract region for merchant. It should use [ISO-3166 Alpha-2] regional code standard. (e.g. US, KR, CN, JP, ALL).
ContactName	1	String	Name of Contact Person
ContactEmail	1	String	Email address of Contact Person
PostalAddress	0..1	String	Postal Address of Contact
PostalCode	0..1	String	Postal Code of Contact
ContactPhone	0..1	String	Phone Number of Contact
ContractDate	1	Date	The date of Merchant Contract
BankInfo	0..1	String	The Banking Information
TaxInfo	0..1	String	The Tax Information

Table 140: ContractInfo Structure

9.8 Enumerations

This section provide the enumerations that used by all the interfaces.

9.8.1 statusCode

Note: the enumerations of statusCode, including but not limited to:

classification	statusCode	Enumeration	Description
	000	success	The request is successfully processed
User related error	101	Invalid UserID	The request is not processed due to the user id is not valid
	102	Existing UserID	The request is not processed due to the user id is already existed
	103	Passwd Error	The request is not processed due to the Passwd of User is error
	104	Invalid Account	The request is not processed due to the Account is not valid
	105	Account type not supported	The request is not processed due to the Account type not supported
Application related error	201	Invalid ApplicationID	The request is not processed due to the ApplicationID is not valid
Developer related error	301	Invalid DeveloperId	The request is not processed due to the DeveloperId is not valid
Default Error	999	Default Error	Default Error

Table 141: stausCode

9.8.2 AppInsStatus Enumeration

Enumeration	Description
0	not installed
1	Installed
2	uninstalled

Table 142: AppInsStatus Enumeration

9.8.3 AppStatus Enumeration

Enumeration	Description
0	Submitted
1	Audited
2	Tested
3	Online

Table 143: AppStatus Enumeration

10.Release Information

10.1 Supporting File Document Listing

Doc Ref	Permanent Document Reference	Description
Supporting Files		
n/a		

Table 144: Listing of Supporting Documents in FOO Release

10.2 OMNA Considerations

This release does not have any OMNA items for handling.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-ER-TAS-V1_0-20150811-A	11 Aug 2015	Status changed to Approved by TP TP Ref # OMA-TP-2015-0123-INP_TAS_V1_0_ERP_for_final_Approval

Appendix B. Use Cases (Informative)

B.1 Built-in IAP Purchase

B.1.1 Short Description

This use case gives an example of the build-in product In Application Purchase.

Build-in product refers to the In application Purchase Item which is already downloaded or installed with the application. When users use a built-in IAP item, they don't need to download other contents or services.

Game A is already downloaded and installed by the user. Game A is free, but its developer embedded several locked levels. Users should purchase before they could enter the locked levels.

The flow a user (Alice) purchases the built-in IAP item is shown below:

1. Alice downloads the Game A from the TAS Enabler.
2. Alice installs the Game A on her device.
3. Alice plays Game A, there are 10 free levels and 3 locked levels in the game. When Alice finished the first 10 levels, she wants to play the last 3 key locked levels. When she tries to enter the locked level, a dialog is shown to the user, "The level is needed to pay $0.99 to be unlocked. If you want to buy it, press OK."
4. Alice presses the OK button. The detailed purchase information is requested to user from the TAS Enabler.
5. The TAS Enabler responds to the application. The price and the service detail information are shown to Alice. A page is shown to let Alice log in TAS Enabler to confirm a TAS payment to make the purchase.
6. Alice confirms to pay.
7. The TAS Enabler returns the payment success response to the application. Then the application unlocks the level to Alice. Now Alice can play the level she just purchased.

B.1.2 Market benefits

If the TAS Enabler provides the IAP function, the developer could embed a store in his application. Built-in IAP product means when users buy the IAP product in the application, they don't need to download extra contents.

B.2 Downloadable IAP Purchase

B.2.1 Short Description

This use case gives an example of the downloadable In Application Purchase.

Downloadable product means the In Application Purchase Item is not downloaded or installed with the application. When the user purchases it and wants to use it, there are some specific content that needs to be downloaded from a content download server.

In this example, the application is a magazine reader, and the magazine is weekly updated. When there is a new magazine published, the developer will update the IAP product information in the TAS Enabler and upload the magazine content to his own magazine download server.

The flow a user (Bob) purchases the newly update magazine is shown below:

1. Bob buys the magazine reader for free from the TAS Enabler.
2. Bob installs the magazine reader on his device.

3. Bob opens the magazine reader and checks the new magazines. The reader sends the request to the magazine download server to check the available magazines.

4. Bob wants to buy the newly updated magazine A. He presses the 'Buy' button. The detailed purchase information is requested from the TAS Enabler.

5. The TAS Enabler responds with the price and the service detail information to Bob. Bob logs in the TAS Enabler to make the purchase.

6. Bob confirms to pay.

7. The TAS Enabler returns the payment success receipt and the content download related information to the application. Then the application transfers the receipt and the download request to the magazine download server.

8. Upon getting the download request and the payment receipt from the application, the magazine download server verifies the payment receipt, then responses the download address to the reader application.

9. The reader application gets the download address and then downloads the content from the magazine download server. After download completely, Bob can read the new magazine in the reader.

B.2.2 Market benefits

If the TAS Enabler provides the IAP function, the developer could embed a store in his application. Downloadable product means the In Application Purchase Item is not downloaded or installed with the application. When the user purchases it and wants to use it, there are some specific content that needs to be downloaded from a content download server.

Appendix C. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

C.1 ERDEF for TAS 1.0 - Storefront Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-TAS-SF-001-M	TAS Storefront	

Table 145: ERDEF for TAS 1.0 Storefront Requirements

C.2 ERDEF for TAS 1.0 - Developer Support Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-TAS-DS-001-M	TAS Developer Support	

Table 146: ERDEF for TAS 1.0 Developer Support Requirements

C.3 ERDEF for TAS 1.0 - TAS Client Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-TAS-C-001-M	TAS Client	

Table 147: ERDEF for TAS 1.0 TAS Client Requirements

C.4 SCR for Storefront

Item	Function	Reference	Requirement
TAS-SF-001-M	Support receiving AppInfoNotify message from Developer Support	Section 9.1.1.1	
TAS-SF-002-M	Support sending AppTransferRequest message to Developer Support	Section 9.1.1.2	
TAS-SF-003-M	Support receiving AppTransferResponse message from Developer Support	Section 9.1.1.3	
TAS-SF-004-M	Support receiving Application Deletion Request message from Developer Support	Section 9.1.2.1	
TAS-SF-005-M	Support sending Application Deletion Response message to Developer Support	Section 9.1.2.2	
TAS-SF-006-M	Support receiving AppUpdateNotify	Section 9.1.3.1	

Item	Function	Reference	Requirement
	message from Developer Support		
TAS-SF-007-M	Support sending AppUpdateRequest message to Developer Support	Section 9.1.3.2	
TAS-SF-008-M	Support receiving AppUpdateResponse message from Developer Support	Section 9.1.3.3	
TAS-SF-009-M	Support receiving Application State Transition Notification message from Developer Support	Section 9.1.4.1	
TAS-SF-010-M	Support sending Application State Transition Response message to Developer Support	Section 9.1.4.2	
TAS-SF-011-O	Support IAP synchronize Request message from Developer Support	Section 9.1.5.1	
TAS-SF-012-O	Support sending IAP synchronize Response message to Developer Support	Section 9.1.5.2	
TAS-SF-013-M	Support receiving AppDownloadRequest message from TAS Client	Section 9.2.1.1	
TAS-SF-014-M	Support sending AppDownloadResponse message to TAS Client	Section 9.2.1.2	
TAS-SF-015-M	Support receiving Malicious Application Report message from TAS Client	Section 9.2.2.1	
TAS-SF-016-M	Support sending Malicious Application Report Response message to TAS Client	Section 9.2.2.2	
TAS-SF-017-O	Support receiving AppFeedbackRequest message from TAS Client	Section 9.2.3.1	
TAS-SF-018-O	Support sending AppFeedbackResponse message to TAS Client	Section 9.2.3.2	
TAS-SF-019-M	Support receiving UserActivationRequest message from TAS Client	Section 9.2.4.1	

Item	Function	Reference	Requirement
TAS-SF-020-M	Support sending UserActivationResponse message to TAS Client	Section 9.2.4.2	
TAS-SF-021-M	Support receiving AppInsStatusReport message from TAS Client	Section 9.2.5.1	
TAS-SF-022-M	Support sending AppInsStatusReportResponse message to TAS Client	Section 9.2.5.2	
TAS-SF-023-M	Support receiving UserAcInfCheckRequest message from TAS Client	Section 9.2.6.1	
TAS-SF-024-M	Support sending UserAcInfCheckResponse message to TAS Client	Section 9.2.6.2	
TAS-SF-025-M	Support receiving AccountInfModifyRequest message from TAS Client	Section 9.2.7.1	
TAS-SF-026-M	Support sending AccountInfModifyResponse message to TAS Client	Section 9.2.7.2	
TAS-SF-027-M	Support receiving AppPurchaseRequest message from TAS Client	Section 9.2.8.1	
TAS-SF-028-M	Support sending AppPurchaseResponse message to TAS Client	Section 9.2.8.2	
TAS-SF-029-M	Support receiving AppUpdateCheckRequest message from TAS Client	Section 9.2.9.1	
TAS-SF-030-M	Support sending AppUpdateCheckResponse message to TAS Client	Section 9.2.9.2	
TAS-SF-031-M	Support receiving AppUpdateDetailRequest message from TAS Client	Section 9.2.9.3	
TAS-SF-032-M	Support sending AppUpdateDetailResponse message to TAS Client	Section 9.2.9.4	
TAS-SF-033-O	Support receiving Application Sorting Request message from TAS Client	Section 9.2.10.1	

Item	Function	Reference	Requirement
TAS-SF-034-O	Support sending Application Sorting message to TAS Client	Section 9.2.10.2	
TAS-SF-035-O	Support receiving Application Search Request message from TAS Client	Section 9.2.11.1	
TAS-SF-036-O	Support sending Application Search Response message to TAS Client	Section 9.2.11.2	
TAS-SF-037-M	Support receiving DeactivationRequest message from TAS Client	Section 9.2.12.1	
TAS-SF-038-M	Support sending DeactivationResponse message to TAS Client	Section 9.2.12.2	
TAS-SF-039-M	Support receiving AppRefundRequest message from TAS Client	Section 9.2.13.1	
TAS-SF-040-M	Support sending AppRefundResponse message to TAS Client	Section 9.2.13.2	
TAS-SF-041-M	Support receiving AppDownloadStatusReport message from TAS Client	Section 9.2.14.1	
TAS-SF-042-M	Support sending AppDownloadStatusResponse message to TAS Client	Section 9.2.14.2	
TAS-SF-043-M	Support receiving AppDetailRequest message from TAS Client	Section 9.2.15.1	
TAS-SF-044-M	Support sending AppDetailResponse message to TAS Client	Section 9.2.15.2	
TAS-SF-045-O	Support receiving Application FeedbackList Request message from TAS Client	Section 9.2.16.1	
TAS-SF-046-O	Support sending Application FeedbackList Response message to TAS Client	Section 9.2.16.2	
TAS-SF-047-M	Support sending Malicious Application Notification message to Developer Support	Section 9.6.1.1	

Item	Function	Reference	Requirement
TAS-SF-048-M	Support receiving Malicious Application Notification Response message from Developer Support	Section 9.6.1.2	
TAS-SF-049-O	Support sending Application Sale Information Notification message to Developer Support	Section 9.6.2.1	
TAS-SF-050-O	Support receiving Application Sale Information Response message from Developer Support	Section 9.6.2.2	
TAS-SF-051-M	Support sending Application State Transition Notification message to Developer Support	Section 9.6.3.1	
TAS-SF-052-M	Support receiving Application State Transition Response message from Developer Support	Section 9.6.3.2	

C.5 SCR for Developer Support

Item	Function	Reference	Requirement
TAS-DS-001-M	Support sending AppInfoNotify message to Storefront	Section 9.1.1.1	
TAS-DS-002-M	Support receiving AppTransferRequest message frome Storefront	Section 9.1.1.2	
TAS-DS-003-M	Support sending AppTransferResponse message to Storefront	Section 9.1.1.3	
TAS-DS-004-M	Support sending Application Deletion Request message to Storefront	Section 9.1.2.1	
TAS-DS-005-M	Support receiving Application Deletion Response message from Storefront	Section 9.1.2.2	
TAS-DS-006-M	Support sending AppUpdateNotify message to Storefront	Section 9.1.3.1	
TAS-DS-007-M	Support receiving AppUpdateRequest message from Storefront	Section 9.1.3.2	
TAS-DS-008-M	Support sending	Section 9.1.3.3	

Item	Function	Reference	Requirement
	AppUpdateResponse message to Storefront		
TAS-DS-009-M	Support sending Application State Transition Notification message to Storefront	Section 9.1.4.1	
TAS-DS-010-M	Support receiving Application State Transition Response message from Storefront	Section 9.1.4.2	
TAS-DS-011-O	Support sending IAP synchronize Request message to Storefront	Section 9.1.5.1	
TAS-DS-012-O	Support receiving IAP synchronize Response message from Storefront	Section 9.1.5.2	
TAS-DS-013-M	Support receiving Application Deletion Request message from Developer's Portal	Section 9.5.1.1	
TAS-DS-014-M	Support sending Application Deletion Response message to Developer's Portal	Section 9.5.1.2	
TAS-DS-015-M	Support receiving Malicious Application Notification message from Storefront	Section 9.6.1.1	
TAS-DS-016-M	Support sending Malicious Application Notification Response message to Storefront	Section 9.6.1.2	
TAS-DS-017-O	Support receiving Application Sale Information Notification message from Storefront	Section 9.6.2.1	
TAS-DS-018-O	Support sending Application Sale Information Response message to Storefront	Section 9.6.2.2	
TAS-DS-019-M	Support receiving Application State Transition Notification message from Storefront	Section 9.6.3.1	
TAS-DS-020-M	Support sending Application State Transition Response message to Storefront	Section 9.6.3.2	

C.6 SCR for TAS Client

Item	Function	Reference	Requirement
TAS-C-001-M	Support sending AppDownloadRequest message to Storefront	Section 9.2.1.1	
TAS-C-002-M	Support receiving AppDownloadResponse message from Storefront	Section 9.2.1.2	
TAS-C-003-M	Support sending Malicious Application Report message to Storefront	Section 9.2.2.1	
TAS-C-004-M	Support receiving Malicious Application Report message from Storefront	Section 9.2.2.2	
TAS-C-005-O	Support sending AppFeedbackRequest message to Storefront	Section 9.2.3.1	
TAS-C-006-O	Support receiving AppFeedbackResponse message from Storefront	Section 9.2.3.2	
TAS-C-007-M	Support sending UserActivationRequest message to Storefront	Section 9.2.4.1	
TAS-C-008-M	Support receiving UserActivationResponse message from Storefront	Section 9.2.4.2	
TAS-C-009-M	Support sending AppInsStatusReport message to Storefront	Section 9.2.5.1	
TAS-C-010-M	Support receiving AppInsStatusReportResponse message from Storefront	Section 9.2.5.2	
TAS-C-011-M	Support sending UserAcInfCheckRequest message to Storefront	Section 9.2.6.1	
TAS-C-012-M	Support receiving UserAcInfCheckResponse message from Storefront	Section 9.2.6.2	
TAS-C-013-M	Support sending AccountInfModifyRequest message to Storefront	Section 9.2.7.1	
TAS-C-014-M	Support receiving AccountInfModifyResponse message from Storefront	Section 9.2.7.2	
TAS-C-015-M	Support sending AppPurchaseRequest message to Storefront	Section 9.2.8.1	

Item	Function	Reference	Requirement
TAS-C-016-M	Support receiving AppPurchaseResponse message from Storefront	Section 9.2.8.2	
TAS-C-017-M	Support sending AppUpdateCheckRequest message to Storefront	Section 9.2.9.1	
TAS-C-018-M	Support receiving AppUpdateCheckResponse message from Storefront	Section 9.2.9.2	
TAS-C-019-M	Support sending AppUpdateDetailRequest message to Storefront	Section 9.2.9.3	
TAS-C-020-M	Support receiving AppUpdateDetailResponse message from Storefront	Section 9.2.9.4	
TAS-C-021-O	Support sending Application Sorting Request message to Storefront	Section 9.2.10.1	
TAS-C-022-O	Support receiving Application Sorting message from Storefront	Section 9.2.10.2	
TAS-C-023-O	Support sending Application Searching Request message to Storefront	Section 9.2.11.1	
TAS-C-024-O	Support receiving Application Searching Response message from Storefront	Section 9.2.11.2	
TAS-C-025-M	Support sending DeactivationRequest message to Storefront	Section 9.2.12.1	
TAS-C-026-M	Support receiving DeactivationResponse message from Storefront	Section 9.2.12.2	
TAS-C-027-M	Support sending AppRefundRequest message to Storefront	Section 9.2.13.1	
TAS-C-028-M	Support receiving AppRefundResponse message from Storefront	Section 9.2.13.2	
TAS-C-029-M	Support sending AppDownloadStatusReport message to Storefront	Section 9.2.14.1	
TAS-C-030-M	Support receiving AppDownloadStatusResponse message from Storefront	Section 9.2.14.2	
TAS-C-031-M	Support sending AppDetialRequest	Section 9.2.15.1	

Item	Function	Reference	Requirement
	message to Storefront		
TAS-C-032-M	Support receiving AppDetailResponse message from Storefront	Section 9.2.15.2	
TAS-C-033-O	Support sending Application FeedbackList Request message to Storefront	Section 9.2.16.1	
TAS-C-034-O	Support receiving Application FeedbackList Response message from Storefront	Section 9.2.16.2	

Appendix D. Informative Flows

D.1 Application Management

D.1.1 Application Upload Flow

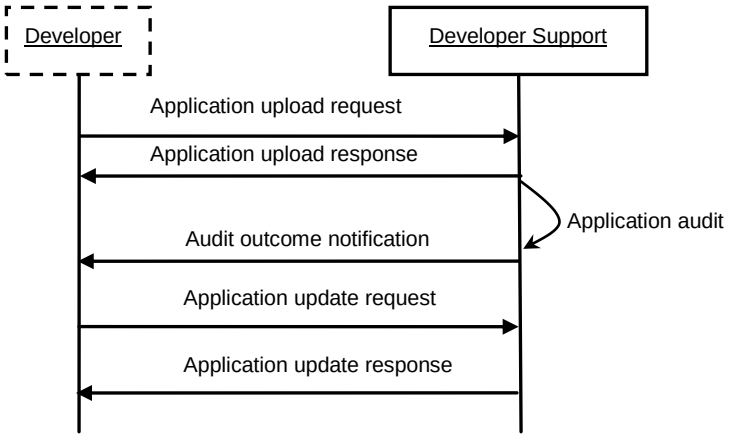

Figure 48: Application Upload Flow

This call flow is triggered by Developer.

1. Developer sends application upload request to Developer Support component, to upload application. The request may contain some contextual information such as:
   - Developer ID
   - Application ID, type, name, description, version
   - Upload time
   - Effective date
   - Expiry date
   - Price
2. Developer Support component sends application upload response to the developer.
3. Developer Support component audits uploaded application and audits related information (e.g. name of the developer) according to the TAS service provider's policies. When to trigger the audit process depends on the Service Provider's policy.
4. Optionally, Developer Support component notifies the developer of the outcome of the audit.
5. Developer sends request to update their own uploaded applications (e.g. submit a new version, including the software and related information) in order to improve the functions or add some new features.
6. Developer Support component returns the application update response to the developer.

Note that Developer Support component also needs to audit the updated applications.

D.1.2 Application Download Flow

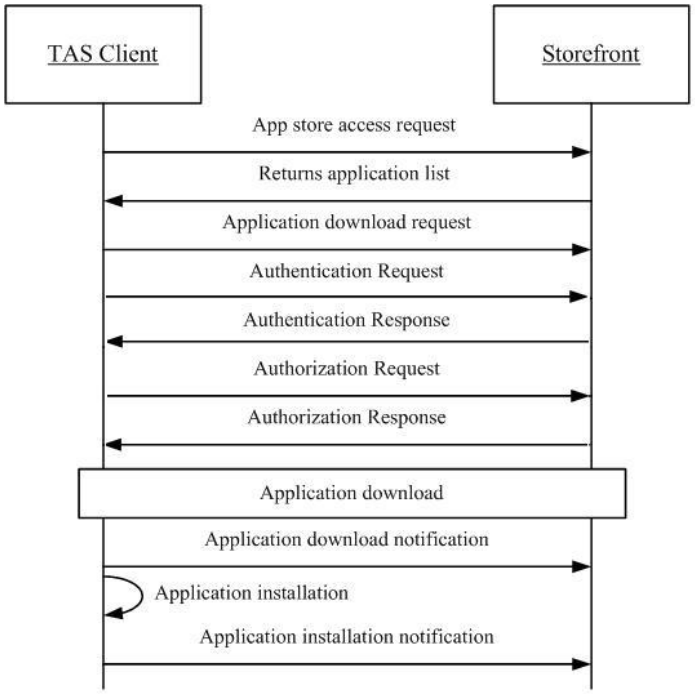

Figure 49: Application Download Flow

This call flow is triggered by TAS Client's internal execution logic.

1. TAS Client sends access request to Storefront, to obtain the application list. The request may contain some contextual information such as:
   - TAS Client ID
   - Application category information
   - Device information, which can be used to select appropriate applications for the device
2. Storefront returns application list to the TAS Client, which is appropriate for the device information.
3. Upon user's selection from the application list, TAS Client sends request to Storefront to download the selected application. The request should contain application's identification information.
4. TAS client sends Authentication Request to Storefront to identify and authenticate the TAS Client.

   Storefront responds with Authentication Response to TAS Client.
5. TAS client sends Authorization Request to Storefront to prevent unauthorized access.
6. Storefront responds with Authorization Response toTAS Client.
7. TAS Client downloads application directly from Storefront, or using other mechanisms, for example, DLOTA.
8. TAS Client sends notification to the Storefront about the application download result.
9. TAS Client installs application on the device.

10. TAS Client sends notification to the Storefront about the application installation result.

Note that the authentication/authorization request and response can apply to other flows as well.

D.1.3 Application Feedback Flow

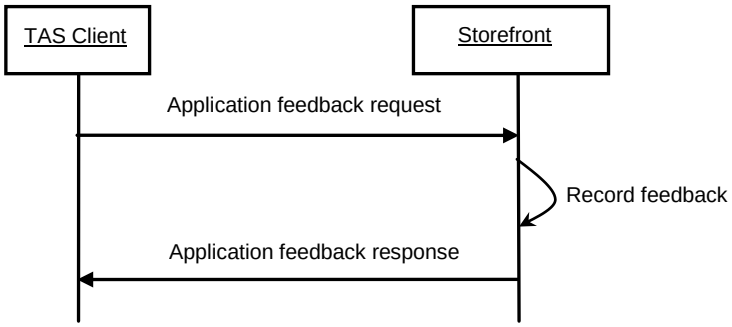

Figure 50: Application Feedback Flow

This call flow is triggered by TAS Client's internal execution logic.

1. TAS Client sends application feedback request to Storefront, to give feedback (e.g. rate and comments) on applications they have used. The request may contain some contextual information such as:
   - User ID
   - Application ID
   - Rate
   - Comment
2. Storefront records the application feedback from TAS Client.
3. Storefront returns the application feedback response to the TAS Client.

D.1.4 Malicious Application Report Flow

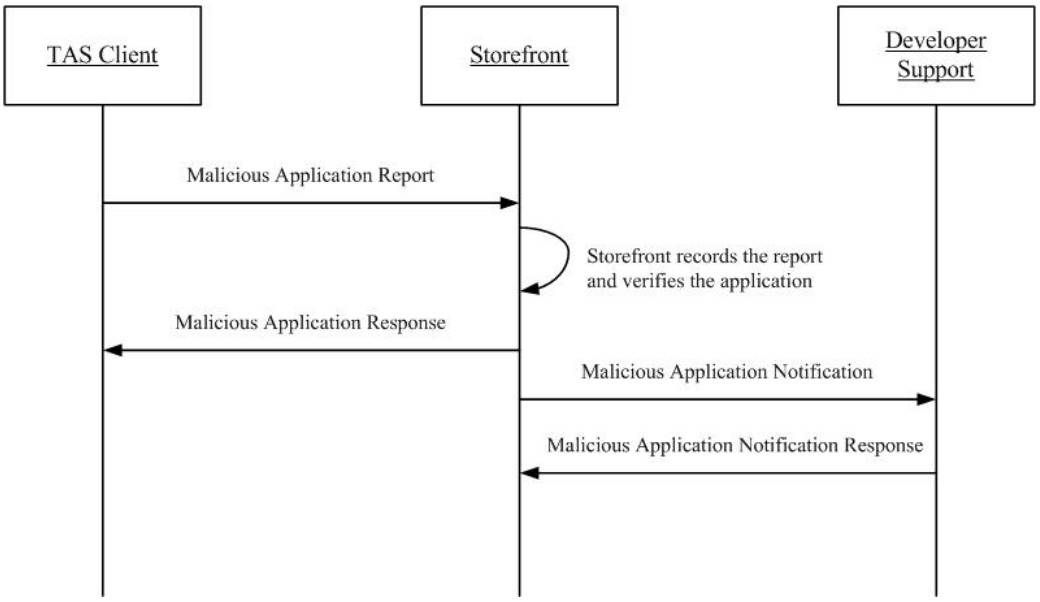

Figure 51: Malicious Application Report Flow

This call flow is triggered by TAS Client's internal execution logic.

1. TAS Client sends Malicious Application Report to Storefront, to provide information on Malicious applications they have used.

   The Report may contain some contextual information such as:

   - User ID
   - Application ID
   - Comment

   (eg: reason to Report this Malicious Application, or suggestion to handle this Malicious Application.)

2. Storefront records the Malicious Application Report from TAS Client and verifies whether the application is malicious according to TAS service provider's policy.

3. Storefront responds TAS Client with Malicious Application Confirm to ascertain this report is correctly received.

4. Storefront sends Malicious Application notification to Developer Support to inform the malicious appplication.

   The Notification may contain some contextual information such as:

   - Application ID
   - Comment

   (e.g. reason why this Malicious Application is identified, suggestion to handle this Malicious Application.)

5. Developer Support responds Storefront with Malicious Application Notification Response to ascertain this notification is correctly received.

D.1.5 Application Gift Flow

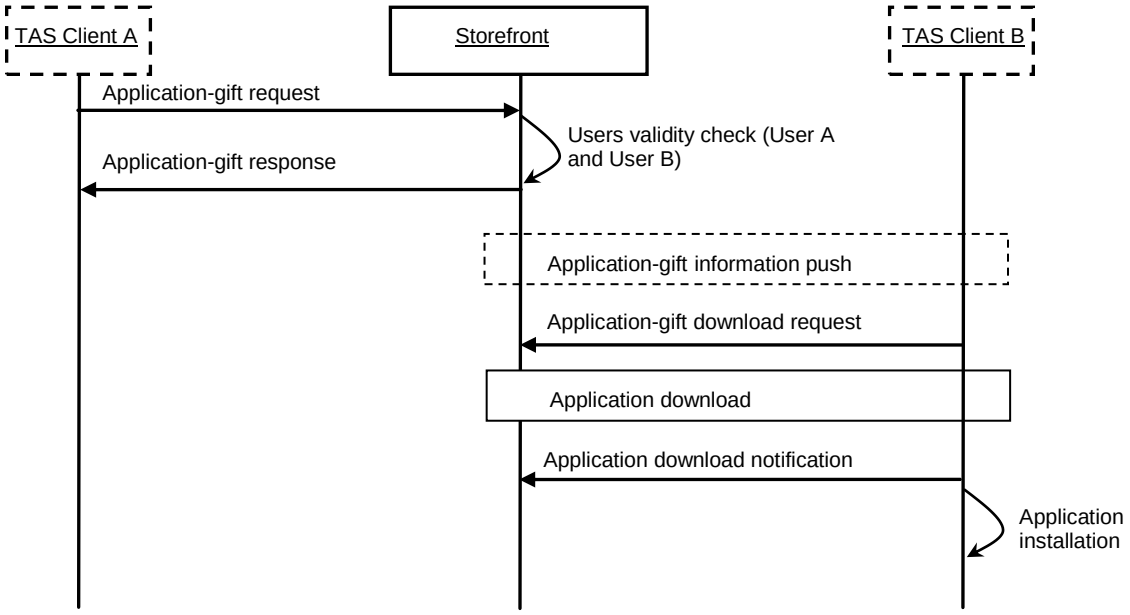

Figure 52: Application Gift Flow

This call flow is triggered by TAS Client's internal execution logic.

1. TAS Client A sends Application-gift request to Storefront. The request may contain some contextual information such as:
   - Application ID
   - User A ID
   - User B ID
2. Storefront checks User's validity(include User A and User B, validation rules are defined by the TAS Service Provider)
3. Storefront sends Application-gift response to TAS client. A
4. Storefront pushes Application-gift information to TAS client B (May via SMS/Email or other ways, This is out of scope of TAS Enabler)
5. TAS Client B sends Application-gift download request to Storefront
6. TAS Client B downloads application directly from Storefront, or using other mechanisms, for example, DLOTA.
7. TAS Client B sends notification to the Storefront about the application download result.
8. TAS Client B installs application on the device. This step is out of scope of TAS Enabler.

D.1.6 Application Begging Flow

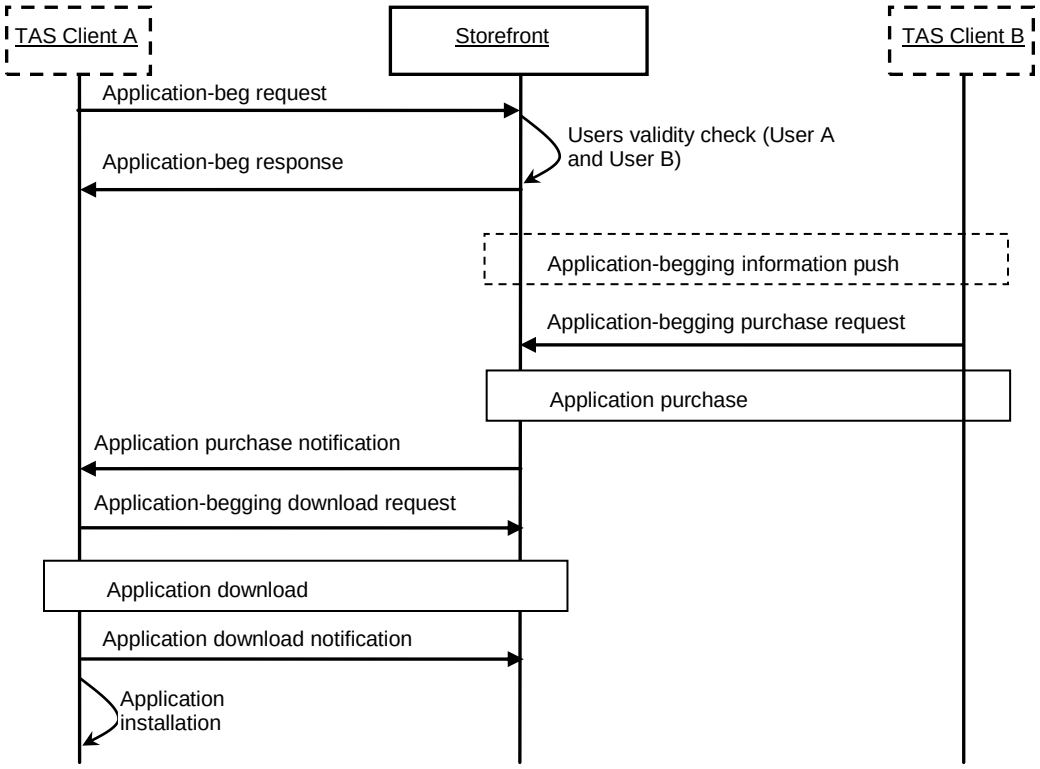

Figure 53: Application Begging Flow

This call flow is triggered by TAS Client's internal execution logic.

1. TAS Client A sends Application-begging request to Storefront. The request may contain some contextual information such as:
   - Application ID
   - User A ID
   - User B ID
2. Storefront checks User's validity(include User A and User B, validation rules are defined by the TAS Service Provider)
3. Storefront sends Application-begging response to TAS client. A
4. Storefront pushes Application-begging information to TAS client B (May via SMS/Email or other ways, This is out of scope of TAS Enabler)
5. TAS Client B sends Application-begging purchase request to Storefront
6. TAS Client B purchase application from Storefront.
7. Storefront sends Application-purchase notification to TAS client. A
8. TAS Client A sends Application-begging download request to Storefront
9. TAS Client A downloads application directly from Storefront, or using other mechanisms, for example, DLOTA.
10. TAS Client A sends notification to the Storefront about the application download result.

11. TAS Client A installs application on the device. This step is out of scope of TAS Enabler.

D.2 User Management

D.2.1 User Activation and modification Flow

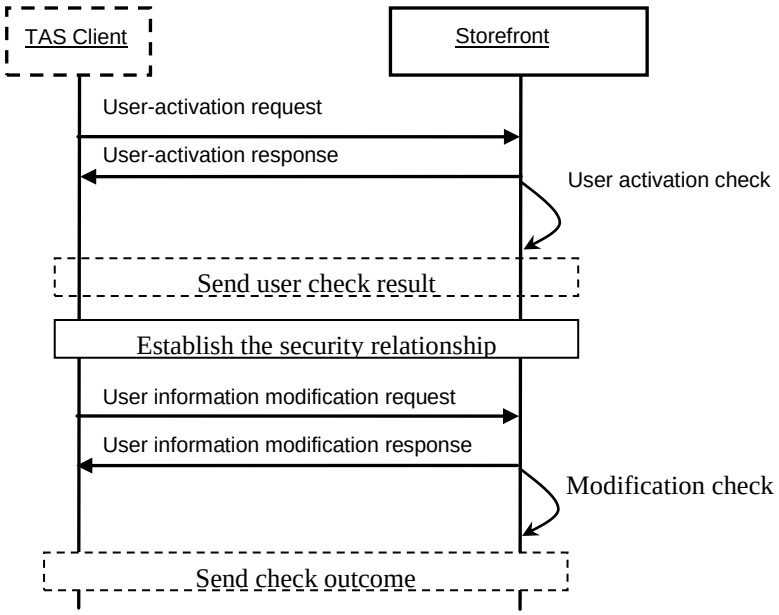

Figure 54: User Accivation and Modification Flow

This call flow is triggered by TAS Client's internal execution logic.

1. TAS Client sends User-activation request to Storefront. The request may contain some contextual information such as:
   - Essential information (e.g. user identifier, name)
2. Storefront sends User-activation response to TAS client.
3. Storefront checks User's validity, validation rules are defined by the TAS Service Provider.
4. Storefront sends check result to User.
5. Storefront establishes the security relationship with the user.
6. TAS Client sends User information modification request to Storefront. The User information modification request may contain the following information(not limited to these):
   - Paid account relation binding modification (e.g. bind Paid account\unbind Paid account\change Paid account)
   - Payment type modification (e.g. post-paid, prepaid)
7. Storefront sends user information modification response to TAS client.
8. Storefront checks User information modification request. Checking criterias are decided by the Service Provider's policy.
9. Storefront sends check outcome to TAS client.