November 2020

Providing Dynamic App Parameter Values Via A Remote Configuration Service

Erik Eldridge

Venkatram Jaltar

Lucas Png

Jon Mensing

Hamid Mousavi

Providing Dynamic App Parameter Values Via A Remote Configuration Service

ABSTRACT

This disclosure describes dynamic management of parameter values for remote configuration of software applications. Per techniques of this disclosure, in addition to serving conventional static values encoded in a template, developers are enabled to define dynamic values in the template that are obtained from a designated service and served at runtime. Declarations are utilized to define a value and/or specify condition syntax in a template to enable local retrieval of static values and retrieval of dynamic values from the designated service. The designated service can be specified by the app developer. The origin of the parameter value is thus made explicit and easily available.

KEYWORDS

- Remote configuration
- Managed value
- App personalization
- App configuration
- A/B testing
- App instance

BACKGROUND

Mobile application (app) and game developers often utilize cloud-based services to specify and/or update app parameters such as color, images, text etc. for apps that run on mobile devices. Cloud-based updating of parameters enables a developer to selectively control the behavior and/or features of an app remotely without having to publish an app update. For example, the look-and-feel of an app, offers, advertising content, etc. can be updated without

requiring the user to download an updated version of the app. Remote configuration is performed by defining a template that includes values for various app settings and fetching the values for those settings (at runtime) from a remote server.

Developers specify settings by defining key-value pairs (parameters and their corresponding values) in the template. The key-value pairs for various app parameters, e.g. color, images, text, etc., are utilized to adjust/configure the app experience.

An initial set of parameter values is utilized to initialize an app and provide a default setting, and parameter values can be dynamically adjusted, e.g. for personalization of the app based on user-permitted factors such as usage of the app, recent interactions with the app, device type, device region/locale, etc. The service that hosts a remote configuration template stores developer-defined key-value pairs for parameter values in the template and serves these to apps upon request. Thus, for a developer that uses the service, it is not possible to use the service to dynamically adjust the parameter values (which limits the range of values that can be served) or to determine the origin of a particular parameter value that is dynamically served to an app instance, when such parameter value is dynamically determined.

DESCRIPTION

Developers benefit from being able to dynamically specify parameter values (rather than static stored key-value pairs). For example, dynamic parameter values can enable developers to run tests where different subsets of users get different parameter values (e.g., A/B testing) and observe the effectiveness of different parameter values. In another example, developers can customize the app experience for different users based on different parameter settings that are determined dynamically.

This disclosure describes a remote configuration service that can dynamically obtain parameters values from external sources and enable developers to determine the origin of dynamic parameter values that are served. Dynamic parameter values may be obtained from services, e.g., an A/B testing service or a personalization service, that provide the values to a remote configuration service that hosts the remote configuration template and serves the parameter values. The remote configuration service can then serve parameter values - both static values stored as key-pairs in the template as well as dynamic values obtained from such other services.

However, if such a service does not store a record of the different dynamic values of parameters that it serves or a history of key-value pairs in the static case, it is not possible for a developer that uses the service to determine the origin of a particular parameter value served to an app instance. This disclosure further enables templates that enable explicit declaration of parameter value management in a template such that it allows developers to determine the origin of parameter values that are served.

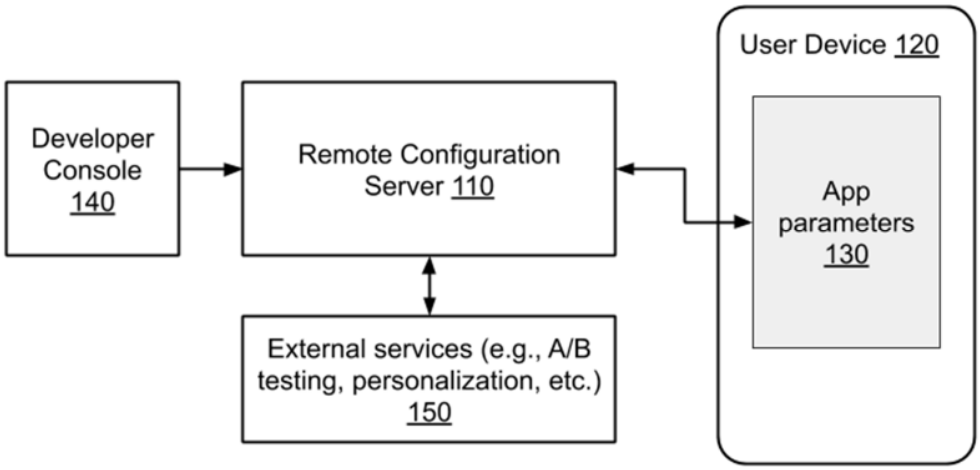

Fig. 1: System environment for dynamic parameter value management

Fig. 1 illustrates an example system architecture for dynamic serving of parameter values for remote configuration of software applications, per techniques of this disclosure. A remote configuration server (110) is utilized to provide app parameters (130) for an app on a user device (120), e.g., in response to a request received from the app. Initialization of app parameters at a time of installation of the app may be performed using on-device default settings declared by the developer or from default settings downloaded from the remote configuration server (which are also provided by the developer). The app may request parameter values at periodic intervals, e.g., upon the occurrence of specific events in the app (e.g., loading a page, completing a game level, etc.).

Developers of the software application can utilize a developer console (140) to specify and/or parameter values, etc. The developer can also utilize external services (150) to obtain the parameter values. For example, such services can include an A/B testing service, a personalization service, etc. With user permission, the dynamic values may be chosen based on information regarding the app instance, e.g., device type, device region/locale, etc.

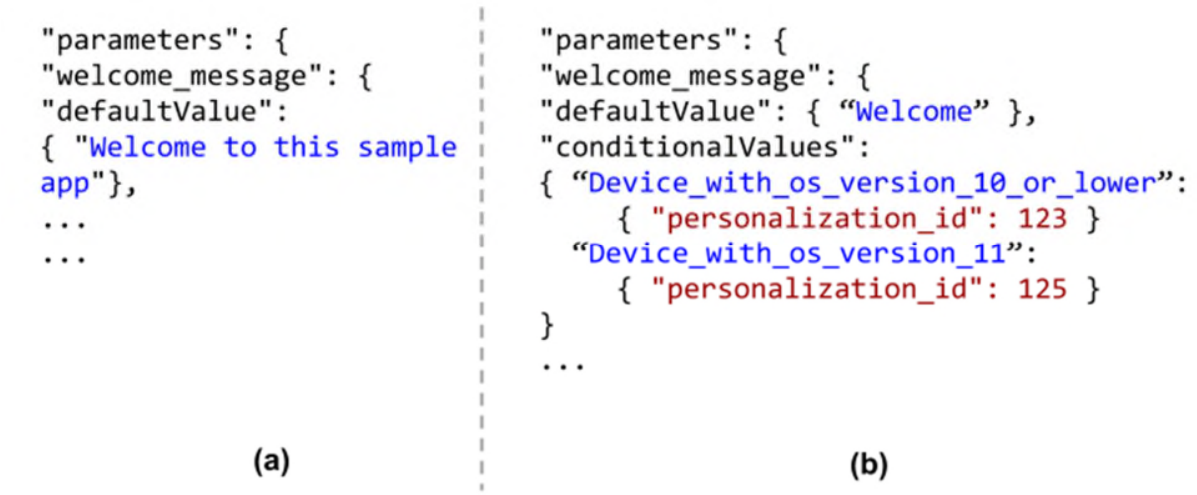

Fig. 2: Parameter declaration; (a) static declaration; and (b) dynamic declaration

Fig. 2 depicts examples of declaration of static and dynamic values for parameters, per techniques of this disclosure. Fig. 2(a) depicts an example static declaration for a parameter ("welcome_message") for which a default value is specified as the string, "Welcome to this sample app." This can be utilized for example, as a message to be displayed when a user first accesses the app.

Fig. 2(b) depicts an example declaration for a parameter that is obtained dynamically, per techniques of this disclosure. The declarations enable defining a value and/or condition syntax in a template to provide management of parameter values External services can be accessed to obtain parameter values, based on the configuration file as defined via a console by a developer.

In this illustrative example, a default value for a parameter ('welcome_message") is specified as "Welcome" and additionally, conditional values are specified (shown in blue). Conditions for the app instance are evaluated (the device operating system version, in this example) and a corresponding parameter value is obtained (from a service that provides parameter values, such as an A/B testing or personalization service) via a dynamic reference parameter ("personalization_id", shown in red) and a value ("123"/ "125") for the parameter identified based on the specified condition. While Fig. 2(b) illustrates obtaining dynamic values conditionally, it is also possible to use dynamic values unconditionally, e.g., by specifying "welcome_message={"personalization_id": 123 }" in the code above. The developer can flexibly define any type of conditions based on user-permitted factors such as user preferences, user activity in the app, etc.

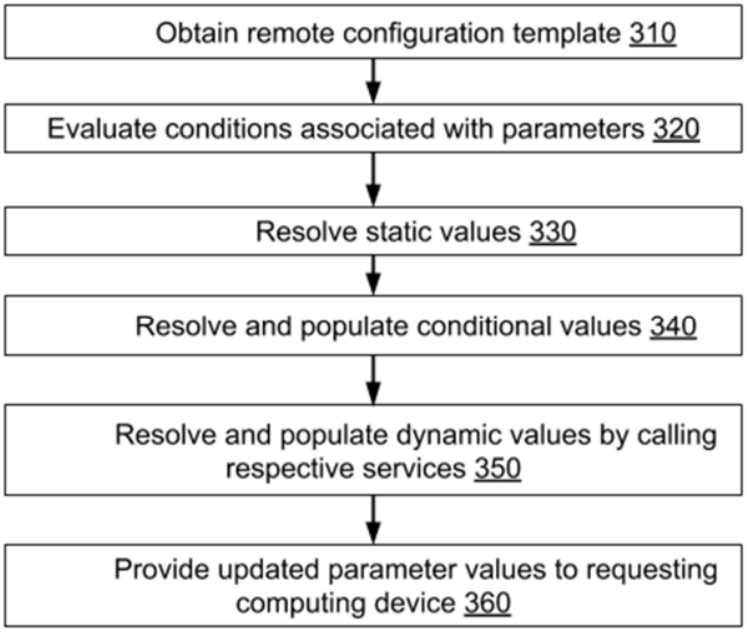

Fig. 3: Management of static and dynamic parameter values

Fig. 3 depicts an example process for the identification and serving of parameter values for a software application via a cloud-based service, per techniques of this disclosure. The process is performed, for example, at a remote configuration server, at the time of request for updated parameters from a software application such as a mobile app or game.

A configuration template is retrieved (310). The template may be retrieved, for example, in response to a request for parameter values from an app on a user device. Conditions associated with various parameters in the template are evaluated (320), e.g., based on user-permitted factors that may be specified in the request. Static values are obtained (330) based on the template. Conditional statements are resolved (340) by evaluating associated rules.

Dynamic values are populated (350) by making a request to a corresponding service that is specified (e.g., by the app developer) as serving the corresponding parameter(s). For example, a request for personalized parameters can be handled by making a request to a designated service that is associated with providing personalized parameters. With reference to Fig. 2(b), such a

request may include the personalization ID. Upon receipt of parameter values from the designated service, the parameter values are provided (360) to the requesting device.

Logging of the parameter values associated with the app instance/impression can be performed on the client-side. External services such as a personalization service or an A/B testing services, etc. may utilize APIs separate from an API associated with the remote configuration service for obtaining data from user device(s). The external services can also maintain a log of dynamically managed values, separate from template values, and can make such values available for developer reference.

This disclosure thus enables an explicit declaration of dynamic values, which enables incorporation of values from an external service, in a way that is easy to discern relative to the other values declared in the template. The condition syntax for the template is available to developers via the console and thus, the developer can determine the origin of a particular parameter value. The origin of the value is thus made explicit and easily available. Also, the actual creation, read, update, and deletion operation of parameter values is performed at the designated service and is under control of the developer, not the remote configuration service. Thus, the parameter values can be updated independent of the remote configuration service.

In the absence of dynamic values as described herein, collisions are impossible because all values for a parameter are declared in the template. However, it remains the developer's responsibility to ensure no collisions occur between the static values in the template and the dynamic values obtained from a designated service.

One approach to avoid collisions is to require the provider of dynamic values to provide key-value pairs back to the configuration service. At this point, there is a risk of collision with the key-value pairs declared in the template. By declaring all keys in the template, and only

allowing providers to define values, this risk is eliminated. For example, if the template declared {welcome_message="hi"} and an external service wanted to incorporate {welcome_message="bye"}, a collision occurs at runtime. However, since “welcome_message” is declared in the template, it is mandatory that the developer choose whether the value is "hi" or <"bye" via the external service> at declaration time. In this manner, collisions can be resolved at declaration, rather than at runtime.

CONCLUSION

This disclosure describes dynamic management of parameter values for remote configuration of software applications. Per techniques of this disclosure, in addition to serving conventional static values encoded in a template, developers are enabled to define dynamic values in the template that are obtained from a designated service and served at runtime. Declarations are utilized to define a value and/or specify condition syntax in a template to enable local retrieval of static values and retrieval of dynamic values from the designated service. The designated service can be specified by the app developer. The origin of the parameter value is thus made explicit and easily available.