Computer Communications 34 (2011) 140–149

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Defining a call control interface for browser-based integrations using representational state transfer

Keith Griffin [a,*], Colin Flanagan [b]

[a] Cisco, Oranmore Business Park, Oranmore, Co. Galway, Ireland
[b] University of Limerick, Limerick, Ireland

ARTICLE INFO

Article history:
Received 2 November 2009
Received in revised form 23 March 2010
Accepted 26 March 2010
Available online 2 April 2010

Keywords:
Open API design
REST
Call control
Interface
Browser

ABSTRACT

While integrating telephony call control signaling into a desktop application is not a new concept, the emergence of web browser-based applications drives the need for feature parity between desktop and browser-based telephony call control applications. Desktop call control applications typically have the advantage of being able to use local protocol stacks or API libraries. A browser-based application does not have the same capabilities by default, especially when running on a constrained device. Traditional telephony call control APIs and architectures do not currently lend themselves for use with web browser-based applications without requiring some form of download which is not desirable. Is it possible to design a call control interface that can be consumed in browser-based applications natively without requiring additional downloads? Representational State Transfer (REST) is a resource oriented architectural style which, when used with HTTP as a uniform interface offers the possibility to use an existing defined set of interfaces to manipulate state transitions for resources. We propose that the REST architectural style can be used with an identified interface design methodology to define an interface for telephony call control capable of being consumed by browser-based applications. Furthermore by utilizing existing industry standards as an example, we suggest that existing standards can be extended to support browser-based integrations using REST.

1. Introduction

The ability to program communications services in IP telephony environments introduced new possibilities for integration systems [1]. It became possible to programmatically access communication services which allowed third party developers to invoke the functionality exposed via published interfaces on a communication system. A third party developer is any developer outside of the organization that developed the communication system and is typically interested in using or extending the functionality of the system within their application. In turn this allowed users of communication systems greater flexibility in their usage of the system. Enterprise telephony systems have been prevalent in business use since the 1970s [2]. Using the PBX as being representative of enterprise communication systems which have evolved through IP telephony to Unified Communications [3–6] it can be seen that the provision of telephony call control interfaces such as Microsoft TAPI, Novell TSAPI, CT-Connect and Sun's Java Telephony API [7–10] can be used by a third party software developer to offer functionality not offered natively by the communication systems itself. An example of a third party communication integration via published system interfaces is the provision of click to call services within a desktop application. Any third party developer can use published call control interfaces from a unified communications system to enable telephony call control within an application. The benefit to the user is that they can control incoming and outgoing calls from within their application rather than using the phone device directly. In some cases unified communications vendors have implemented their own integrations based on their own open interfaces demonstrating that the same open interfaces can be used natively in addition to third party use [11,12]. Cleary these interfaces and the applications and feature extensions that they offer have value, however the interfaces are predominantly aimed at desktop client applications where a local protocol stack or other local component can be assumed. What if integration with a web browser-based application is required? Furthermore, what if the browser-based application is running on a constrained device? While a browser-based application running on a computer desktop might have the resources available to work-around a given interface by supporting a local protocol stack or other downloaded component, the same cannot be assumed for a browser-based application running on a constrained device. Representational State Transfer (REST) [13], is

* Corresponding author. Tel.: +353 87 650 4145.
E-mail addresses: kegriffi@cisco.com (K. Griffin), colin.flanagan@ul.ie (C. Flanagan).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.03.029

a hybrid architectural style for distributed hypermedia systems which uses a constrained interface to manipulate the state of any given resource. In this paper, we propose that the principles of a REST based system can be used to define a telephony call control interface suitable for consumption in browser-based communication integrations. Furthermore, we propose that this interface can be used without requiring any other local client software component other than a HTTP capable browser. Additionally, we identify and apply a methodology suitable for the design of such interfaces.

The paper is organised as follows. First, we present the necessary background and state of the art describing the problem domain of telephony call control in browser-based integrations. We introduce the principles of REST in the background section. Next, we look at the interface design by reviewing an existing telephony call control standard and outlining a methodology for REST based interface design, before applying this methodology to define our proposed interface. After presenting the proposed interface design, we evaluate the interface by applying a common call control use case. Finally, we conclude by providing the reader with an overview of our findings and outline future work.

2. Background

2.1. Enterprise telephony call control

Computer Telephony Integration (CTI) is based on the exchange of messages between computer and telephony systems. It is not a new concept, with implementations in the industry available since the 1980s. It is however an important concept in the enterprise communications environment and continues to be a commonly used technology in modern deployments. While CTI covers a broad range of specific integrations this research considers desktop based call control as it can be related to browser-based call control. In this context, CTI refers to the signalling associated with the call, hence the term call control. CTI does not imply control of the media associated with the call such as the voice path or the media termination of the voice-path. Later, we will define a representative subset of call control functionality which will be used to test the assertion that call control interfaces can be defined for use within a browser-based application. While introducing telephony integration in a browser-based environment using REST is a new paradigm it does not change the features of a telephony system. Therefore, rather than creating yet another CTI interface, we identify an established and accepted industry standard as a basis for the interface design. The standard chosen is Computer Supported Telecommunications Applications (CSTA) which is an accepted and current industry standard [14]. CSTA defines interfaces related to all aspects of enterprise telephony including an interface called "Call Control Services and Events". CSTA is chosen as an example of an independent, well formed and progressive interface. Independent, as the standard is driven by open working groups without single vendor influence. Well formed, as the standard has been through three major phases of revision and progressive as the standard continues to evolve and considers new forms of interface presentation such as SIP-CSTA [15].

2.2. Browser-based communication integration

Having introduced the concept of a communications integration system it is necessary to focus on one particular consumer of communication integration system interfaces which are not well served by current interface implementations, namely browser-based web clients. Call control CTI clients are typically installed with all the required run-time software components to facilitate bi-directional communication between the client and server in real-time. A study of the third party CTI client implementations previously identified showed that remote procedure call (RPC) [16] and remote method invocation (RMI) [17] between the client and server are common approaches. While these approaches appear to work well for installed desktop clients, a different approach beyond the current state of the art is required for browser-based web clients on constrained devices. Technically it would be possible to take a client side component such as a protocol stack and embed it in a web client in the form of a once off or infrequent download. Depending on the size and nature of the download this may or may not be acceptable for browser-based deployment. Several factors may influence the acceptability of a component download ranging from IT policy to performance to security. Technical factors include the basic ability of a device to accept a component download or handle client side call control processing. For the purpose of this research we introduced a technical constraint. The constraint is that *no client side download will be permitted apart from the browser itself*. In addition to this constraint, no extensions to the browser or browser plug-in component may be used. It is intended that by placing this constraint on the research that the output will be applicable to a range of scenarios for fixed and mobile computing where only a web browser is available to consume a communications integration interface such as CTI call control. It has been shown that automated call handling and user settings can be addressed via REST [18] but these synchronous operations based on user and device settings differ from real-time communication interfaces as in the case of telephony call control. Also, some communications integration interface definitions like CSTA, have been extended to offer SOAP based Web Services [19]. While such an approach is widely accepted for desktop browser-based environments it will break the constraint outlined above as it would involve the download of the Web Service Definition Language (WSDL) and/or the use of a local SOAP stack in the browser-based client application or as a plug into the browser. In order to work within our constraint the browser-based communication integration mechanism chosen must work within the constraints of the browser itself. A safe way to do this is to work strictly within the constraints of the standards that a web browser implements, in particular HTTP [20], thus extending the reach of telephony call control systems to clients that cannot be addressed currently.

2.3. An introduction to representational state transfer

Representational State Transfer or REST as it has become commonly known, is a hybrid architectural style for distributed hypermedia systems. It is not dependent upon any particular protocol but typically uses HTTP. This means that it is possible to create a REST based system that is not built upon HTTP; however, most practical implementations of REST are built using HTTP. REST is not a standard however it does prescribe the use of standards, including but not limited to HTTP, URI, XML and HTML.

2.3.1. REST design principles

The following are design principles of a REST based system.

- Stateless: Every request from client to server must contain all the information required to execute the request and must not rely on information known to the server.
- Uniform interface to support state transfer consisting of:
  - A constrained set of well defined operations e.g. the HTTP methods GET, PUT, POST, DELETE.
  - A constrained set of content types e.g. text/xml, image/jpeg.
- Client server pull interaction: consuming clients pull representations.
- Uniquely Named Resources: A URI is used to name the resources which comprise the system [21].