IHAL AND WEB SERVICE INTERFACES TO VENDOR CONFIGURATION ENGINES

John Hamilton, Timothy Darr, Ronald Fernandes
Knowledge Based Systems, Inc.

Joe Sulewski
L-3 Telemetry East

Charles Jones
812TSS/ENTI
Edwards AFB, CA

ABSTRACT

In this paper, we present an approach towards achieving standards-based multi-vendor hardware configuration. This approach uses the Instrumentation Hardware Abstraction Language (IHAL) and a standardized web service Application Programming Interface (API) specification to allow any Instrumentation Support System (ISS) to control instrumentation hardware in a vendor neutral way without requiring non-disclosure agreements or knowledge of proprietary information. Additionally, we will describe a real-world implementation of this approach using KBSI's InstrumentMap application and an implementation of the web service API by L-3 Communications Telemetry East.

KEYWORDS

IHAL, Instrumentation configuration, metadata, web services, XML, API

INTRODUCTION

Current Instrumentation Support Systems (ISSs) are required to use vendor-specific formats to configure instrumentation systems prior to testing. Since there are a number of vendors selling various data acquisition, control, transmission and storage components, an ISS must be programmed to support all of the specific formats required to interface with these various hardware and software systems. Since it is practically impossible for a single ISS to interface with every known system and vendor, the ISS developers typically select only a few and offer solutions only for those hardware vendors. Similarly, range customers often develop their own software around a single product line, effectively locking them into that line. This prevents the ability to mix and match vendors, resulting in the lack of a robust and cost-effective solution. Moreover, hardware vendors typically supply their own configuration software that works only with their own products. Thus, an instrumentation engineer who is trained on one vendor's

software must learn a new software system in order to use a different vendor's hardware, raising the cost of switching vendors, reducing competition, and making interoperability almost impossible.

The Instrumentation Hardware Abstraction Language (IHAL) and subsequent updates were introduced at ITC in 2006 and 2008 respectively [1][2]. By offering a standard, XML-based, vendor-neutral way to describe and configure instrumentation hardware, IHAL is a major step toward vendor interoperability. IHAL explicitly represents all of the common instrumentation settings, and provides generic constructs that allow vendors to expose their vendor-specific settings as well. By utilizing a standard language such as IHAL, an ISS needs to understand only a single representation in order to support all IHAL-capable hardware vendors. This enables instrumentation engineers to be trained on only one IHAL-aware application and be able to configure multi-vendor instrumentation systems with little or no additional training.

NEED FOR A COMMON API

Relying only on a common language such as IHAL requires that hardware vendors expose in the IHAL format every detail of their hardware's configuration parameters that is necessary to create a valid configuration. Some of this information, such as the way configurable attributes interact with each other, is often proprietary and usually encapsulated in the vendor's configuration software (or in a third-party ISS under a non-disclosure agreement). Without this information, vendor-neutral configuration software would require the user go through the tedious process of building a configuration, loading it into the vendors' software for validation, fixing errors, and then re-loading the modified configuration until all errors are resolved.

This exposure of proprietary information can be avoided if vendors provide access to their internal validation engines through an application programming interface (API). Having an API allows third-party software to interact directly with the vendor's configuration software without requiring usage of the vendor's software user interface. Further, a third-party application could request individual setting changes to a configuration model in the vendor software and receive immediate feedback about the validity of these changes. If such an API were standardized so that all vendors implemented the same basic API function calls, then a third-party ISS could be developed that is capable of configuring the hardware of any vendor who implements the standard API.

When defining the specification for a standard API, interoperability is maximized if implementations of the API are not tied to a specific programming language. Such language-independence can be achieved by implementing the API as a web service, which allows disparate applications to exchange information using the common Hypertext Transfer Protocol (HTTP). Language-independence is the primary reason why this approach is preferred over the other popular approach to achieving inter-process communications: Remote Procedure Call (RPC). Most implementations of RPC are highly-dependent on the specific language or technology used to develop them, reducing interoperability by confining clients to a specific technology. The benefits of utilizing web services in future T&E environments are discussed in detail in [3].

THE IHAL CONFIGURATION API

To address these interoperability issues, we have designed a web-services-based API for querying and configuring vendor systems using IHAL as the communication language. This API specification consists of only five basic functions, described in detail in subsequent subsections.

The IHAL configuration API was designed to be implemented as a Representational State Transfer (RESTful) web service. The decision to use REST as opposed to the Simple Object Access Protocol (SOAP) was made in order to avoid the increased complexity of SOAP. SOAP messages must be wrapped in a specifically-formatted header. This additional markup makes messages larger and requires more time to transmit. In practice, the rigidity of SOAP message formats typically requires a special toolkit in the development environment to program. The main reason for this additional rigidity is to achieve strongly-typed parameters. For our purposes, parameter type-checking is less important because all parameters and responses will be valid IHAL text that conforms to the IHAL XML schema.

API Function 1: Retrieving the Vendor's Pool

Although this represents a single function, the API specification defines four URLs the web service must implement depending on which portion of the pool to return:

- *http://<host>:<port>/pools/card* to retrieve the card pool
- *http://<host>:<port>/pools/instrument* to retrieve the instrument pool
- *http://<host>:<port>/pools/transducer* to retrieve the transducer pool
- *http://<host>:<port>/pools/complete* to retrieve all pools

This function is called anytime an up-to-date list of IHAL pool specifications is needed. As described in [1], the IHAL "pool" contains spec-sheet-style information about all of the hardware available for use. The pool includes descriptions of each device's functions, physical characteristics, and configurable parameters.

The vendor web service returns a complete IHAL pool specification, rooted at the <ihal> element, and containing some combination of <instrumentPool>, <transducerPool>, <functionPool>, and <cardPool> child elements. This function should be called using the HTTP "GET" verb.

API Function 2: Retrieving the Current Configuration

The URL for accessing this function is *http://<host>:<port>/configurations/current*.

This function is called initially when the end-user configuration software loads to get a complete IHAL file containing the current hardware and settings loaded in the vendor's system. Called using the HTTP "GET" verb, this function requires no input, and returns a complete IHAL specification, including both the configuration as well as the pool items for every device in the configuration.

API Function 3: Modifying an Existing Configuration

The URL for accessing this function is *http://<host>:<port>/modifications/new*.

This function is called whenever a user enters or changes a configuration setting. The modified settings are passed to the web service as a string using an HTTP "POST" parameter named *ihal*. This parameter contains an IHAL file consisting of only a <configuration> section with one or more <*Use> elements each containing one or more <setParameter> elements. All ID references are referencing IDs contained in the pool specification initially returned by the vendor's web service (either through a call to retrieve the pool or to retrieve the current configuration). An example input is given in Listing 1. In this example, the user is changing the value of one setting to 10.0 dB.

```
<ihal:ihal xsi:schemaLocation="http://www.kbsi.com/IHAL /IHAL_3_007/ihal3.xsd"
xmlns:ihal="http://www.kbsi.com/IHAL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ihal:configuration id="config0">
    <ihal:name>Any configuration name</ihal:name>
    <ihal:network id="network0">
      <ihal:cardUse poolRef="poolID0" id="card31">
        <ihal:setParameter parameterIDRef="poolID0-channel1-gain">
          <ihal:setConfigurableNumericParameter>
            <ihal:value>10.0</ihal:value>
            <ihal:units>dB</ihal:units>
          </ihal:setConfigurableNumericParameter>
        </ihal:setParameter>
      </ihal:cardUse>
    </ihal:network>
  </ihal:configuration>
</ihal:ihal>
```

Listing 1: Example value for the *ihal* parameter.

The return value of this function is a string containing the "impact" of the modification. This impact is a partial IHAL configuration containing the new settings for everything that has changed. This partial IHAL file will contain only a <configuration> section with one or more <*Use> elements, each containing one or more <setParameter> and/or <restrictedParameter> elements. The <restrictedParameter> elements are used to restrict the set of valid values for a given parameter. An example return value is shown in Listing 2. In this example, the vendor software confirms the changing of one setting to 10 dB (via the <setParameter> element), and also restricts the possible values of a different setting to the range -0.5 to +0.5 (via the <restrictedParameter> element).

```
<ihal:ihal xsi:schemaLocation="http://www.kbsi.com/IHAL /IHAL_3_007/ihal3.xsd"
xmlns:ihal="http://www.kbsi.com/IHAL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ihal:configuration id="config0">
    <ihal:name>Any configuration name</ihal:name>
    <ihal:network id="network0">
      <ihal:cardUse poolRef="poolID0" id="card31">
        <ihal:setParameter parameterIDRef="poolID0-channel1-gain">
          <ihal:channelNumber>0</ihal:channelNumber>
          <ihal:setConfigurableNumericParameter>
            <ihal:value>10.0</ihal:value>
            <ihal:units>dB</ihal:units>
          </ihal:setConfigurableNumericParameter>
```

```
        </ihal:setParameter>
        <ihal:restrictedParameter parameterIDRef="poolD0-channel1-offset">
          <ihal:channelNumber>0</ihal:channelNumber>
          <ihal:configurableNumericParameter>
            <ihal:minimumValue>
              <ihal:value>-0.5</ihal:value>
            </ihal:minimumValue>
            <ihal:maximumValue>
              <ihal:value>0.5</ihal:value>
            </ihal:maximumValue>
          </ihal:configurableNumericParameter>
        </ihal:restrictedParameter>
      </ihal:cardUse>
    </ihal:network>
  </ihal:configuration>
</ihal:ihal>
```

Listing 2: Example "impact" returned by a configuration modification.

The return of this impact is important because it provides the "hook" into the vendor's configuration logic without revealing the underlying rules. Returning this information to the client application ensures not only that the configuration it is displaying to the user is valid, but also that it is correctly restricting the possible values for all configurable parameters.

API Function 4: Creating a New Configuration in the Vendor's System

The URL for accessing this function is: *http://<host>:<port>/configurations/new*

This function is called when a new configuration is built in a client application and needs to be "pushed" to the vendor's software. This function works similar to the call to modify a configuration (see previous subsection) except that this call creates a new configuration rather than modifying an existing one. When the vendor service receives this call it should generate a new configuration using the IHAL elements associated with the <configuration> element to populate items such as name, date, etc, and the IHAL <*Use> elements to populate the hardware and settings contained in this configuration.

The input to this function is a single parameter called *ihal* that is passed using the HTTP "POST" verb. The value of the *ihal* parameter is a string containing a complete IHAL configuration, rooted at the <ihal> element and containing at least one <network> element populated with at least one <*Use> element. This parameter should not contain any pools. Thus, any call to this function should be preceded at some point by a call to the "Retrieving a Vendor's Pool" function.

If any of the settings passed in the new configuration are invalid, the vendor tool will automatically choose new setting values that form a valid configuration. The return value of this function is a string containing a partial IHAL configuration representing any settings that changed from the original input. This partial IHAL file will contain only a <configuration> section with one or more <*Use> elements, each containing one or more <setParameter> and/or <restrictedParameter> elements. The <restrictedParameter> elements are used to restrict the set

of valid values for a given parameter. This is identical to the type of value returned by a call to modify an existing configuration (see previous subsection).

API Function 5: Programming the Hardware

The URL for accessing this function is *http://<host>:<port>/programRequests/<IHAL XML ID>*.

This function is called once a valid configuration is established, to actually program a specific device or devices. The device(s) to program is specified in the URL itself using the corresponding "id" attribute for the IHAL file. The *<IHAL_XML_ID>* portion of the URL must be the value of the "id" attribute for one of the following IHAL elements:

- <instrumentUse> or <cardUse>, to program that specific DAU or card.
- <network>, to program all devices in that network
- <configuration>, to program all devices in all networks in that configuration.

This function returns one of the following status strings:

- **VALID**, if both the id and current configuration are valid, and the hardware can be programmed
- **INVALID_ID**, if the id cannot be found or if it is not the id of an appropriate element.
- **INVALID_CONFIG**, if the id can be found, but the current configuration is not valid
- **INVALID**, for all other errors

DEMO WITH INSTRUMENTMAP AND VISTA TEC

In order to validate the API, we prepared a proof-of-concept demonstration using KBSI's InstrumentMap application as the ISS and L-3 Communications' VistaTEC application as the vendor's configuration system. For the demonstration, L-3 implemented the IHAL configuration API on top of their existing VistaTEC software, which is the program currently used to configure their hardware. KBSI added support for calling the API functions to the InstrumentMap application. This enabled the configuration of L-3 hardware from within InstrumentMap, using only valid IHAL text as the representation language.

To configure the demonstration, we first selected a small representative set of instrumentation hardware from L-3's analog product lines and modeled each device's pool-level attributes in IHAL. This step involved mapping L-3 nomenclature to IHAL nomenclature, and using IHAL's custom 'generic' parameters to represent those settings and attributes which do not map directly to IHAL named concepts. The hardware chosen to best illustrate the demonstration concepts consists of a single data acquisition unit (DAU) and 3 signal conditioning cards. Using this setup, we were able to successfully demonstrate four scenarios, which are described in detail below.

Demo Scenario 1

For the first demonstration, VistaTEC is pre-configured with a DAU containing three signal conditioning cards. In the first step of the demonstration, InstrumentMap issues a request for the current configuration to VistaTEC, which returns a complete IHAL file. This file is then loaded and displayed in the InstrumentMap user interface.

Next, the user makes a change to a configurable parameter in InstrumentMap. This parameter value change is then sent to VistaTEC as a partial IHAL file. VistaTEC processes the change in real-time and returns a partial IHAL file that represents the 'impact' of this change.

For example, when the user increases the gain on a particular card, the impact returned by VistaTEC through the API consists of 1) The new gain value, confirming that it was changed, 2) A new offset value, indicating that this value had to change in order to support the increased gain, and 3) A new, narrower range of valid offset values, indicating that this was also affected by the increased gain. This scenario is illustrated in Figure 1.

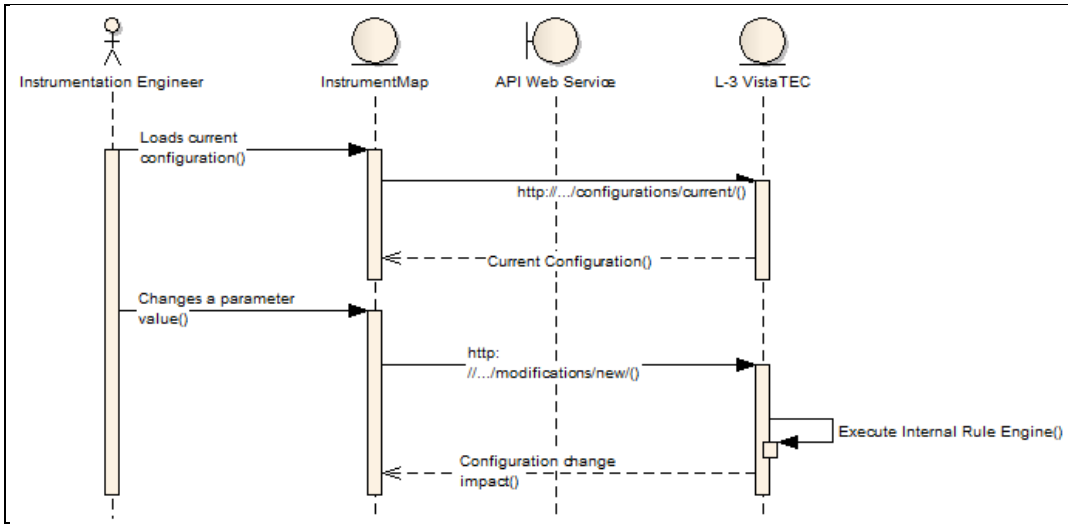

Figure 1: Sequence of Scenario 1

Demo Scenario 2

The second scenario involves allowing the InstrumentMap user to retrieve only the pool from the vendor's web service and then build a configuration from scratch in InstrumentMap. This new configuration can then be "pushed" to the web service where it will be automatically loaded into VistaTEC. The significance of this extension is that users can build scenarios from within a user interface of their choice, regardless of their choice of hardware vendor. This scenario is illustrated in Figure 2.

Demo Scenario 3

The third scenario demonstrates how a single client (InstrumentMap) can connect to multiple servers (VistaTEC) and configure hardware on both servers from a single location. The significance of this scenario is that multiple web services can be deployed in a variety of different architectures. For instance, a multi-vendor environment may consist of one web service for each vendor, or each DAU may host its own web service for configuring only the cards attached to it. All of these web services can now be accessed and maintained from a single user interface as though they were a single configuration. This scenario is illustrated in Figure 3.

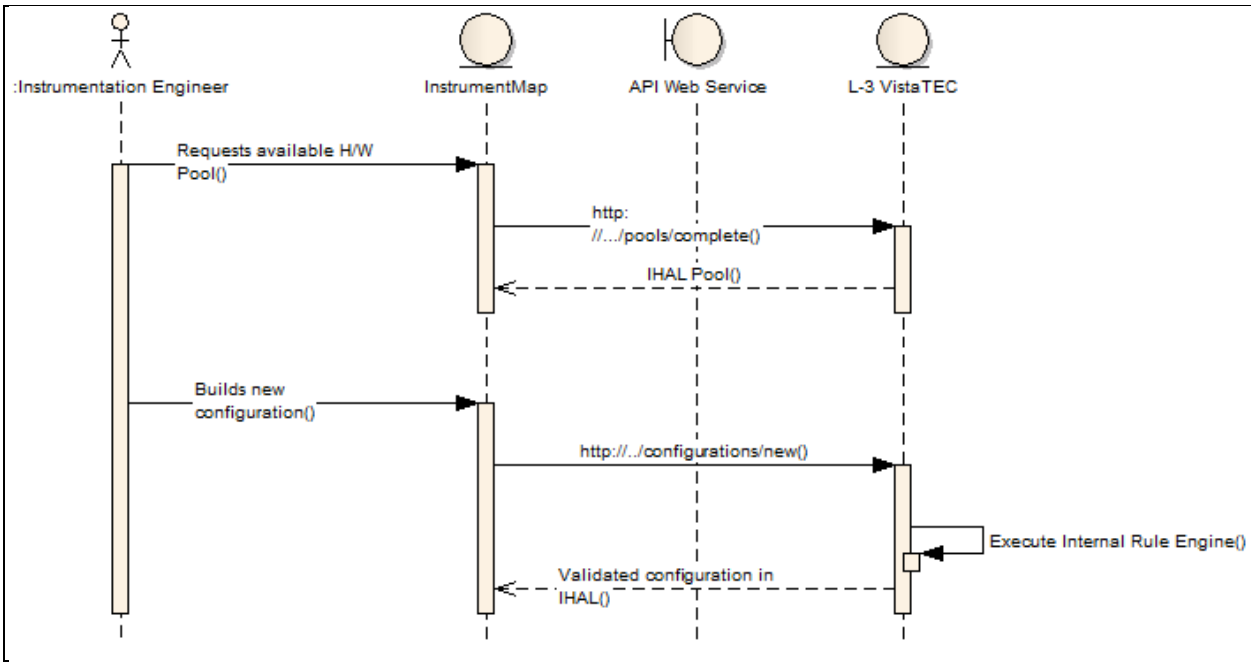

Figure 2: Sequence of Scenario 2

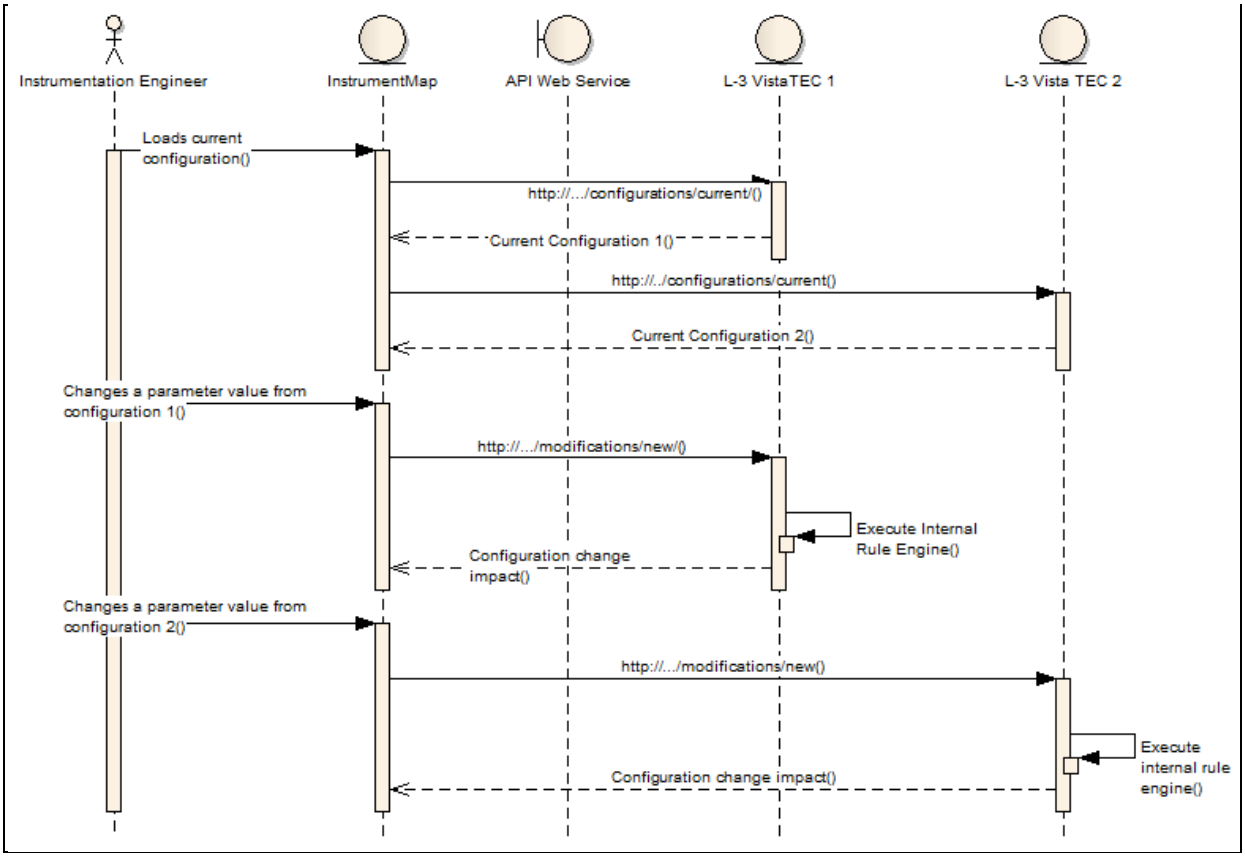

Figure 3: Sequence of Scenario 3

Demo Scenario 4

The fourth and final scenario involves connecting multiple clients (InstrumentMap) to a single web service (VistaTEC). In this instance, each client can work on a different portion of the configuration, and the up-to-date configuration can be retrieved from the web service at any time through a simple "refresh." The significance of this demo is that multiple engineers can configure different parts of a large instrumentation network that is hosted on a single web service and remain in sync with each other. This scenario is illustrated in Figure 4

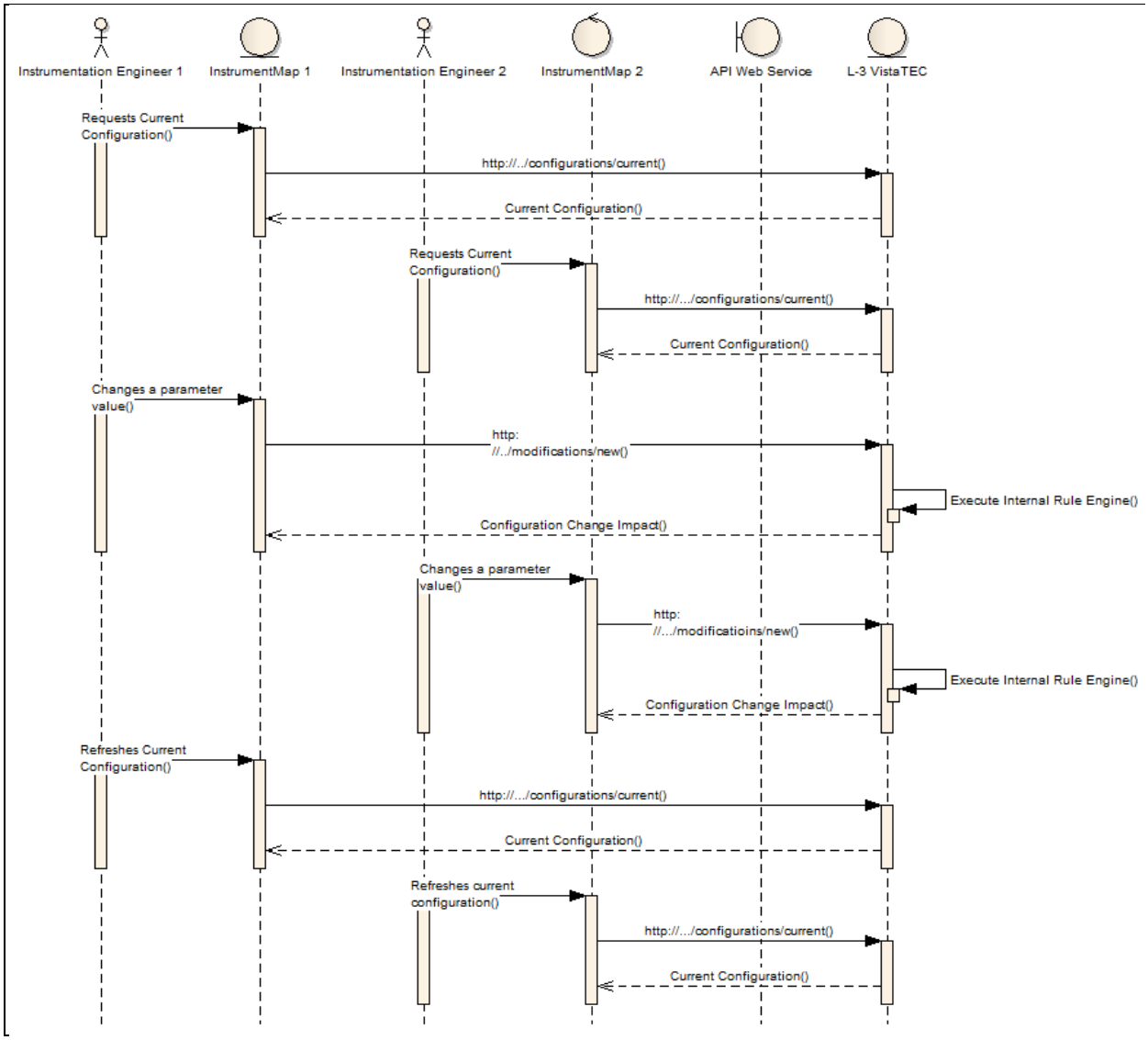

Figure 4: Sequence of Scenario 4

FUTURE WORK

The current IHAL and associated API provide all the mechanisms required to change the values of instrumentation settings and receive the impact in a vendor-neutral way. Hence, any aspect of

instrumentation configuration that can be reduced to a finite set of settings can be achieved using this approach. However, by adding just a few additional functions to the API and constructs to the language, the complexity of some tasks can be reduced.

One example of this is the design and configuration of the PCM stream. While it should be possible to reduce this task to a finite (but large) set of "settings", it would reduce complexity to add a small set of functions to the API for specifically configuring the PCM stream. Additionally, since the Telemetry Attributes Transfer Standard (TMATS) already exists for defining PCM streams, it may be useful to use a combination of IHAL and TMATS as the messages for these functions. However, this task would still follow the same configuration scenario described in this paper: The client sends a vendor-neutral configuration request to the vendor's service, and receives back an "impact" that includes the changes necessary to make the configuration valid.

Another enhancement to the web service that would increase usability would be the establishment of a client "session" with a publish/subscribe mechanism so that the vendor web service can push updates to all subscribing clients. This would eliminate the need for clients to occasionally poll (i.e. "refresh") the web service to receive updates. We are currently looking at various standard protocols for achieving this over HTTP, such as the Bayeux protocol, described in [4].

CONCLUSIONS

Even with the use of a common language for configuring instrumentation hardware such as IHAL, true multi-vendor hardware configuration cannot be achieved without either requiring that vendors reveal their proprietary internal validation rules or allowing them to expose their functionality through a common API. We have developed a simple web-services based API specification consisting of 5 basic functions that, combined with IHAL, allows a single user interface to be used to configure multiple vendors' hardware. We have also validated the API by developing a proof-of-concept demonstration using KBSI's InstrumentMap software and L-3 Communication's VistaTEC.

REFERENCES

[1] Hamilton, Fernandes, Koola, and Jones, *An Instrumentation Hardware Abstraction Language,* Proc. International Telemetering Conf., Vol. XXXXII, (2006) Paper 06-10-02, San Diego, CA.

[2] Hamilton, Fernandes, Graul, Darr, and Jones, *Extensions to the Instrumentation Hardware Abstraction Language (IHAL),* Proc. International Telemetering Conf., Vol. XXXXII, (2008) Paper 08-19-04, San Diego, CA.

[3] Sulewski, Hamilton, Darr, and Fernandes, *Web Service Applications in Future T&E Scenarios*, Submitted to the International Telemetering Conf. (2010), San Diego, CA.

[4] Russell, Alex; Wilkins, Greg; Davis, David;Nesbitt, Mark. The Bayeux Protocol. The Dojo Foundation. 2007. URL:http://svn.cometd.com/trunk/bayeux/bayeux.html. Accessed: 2010-04-22.