IVI Interchangeable Virtual Instruments

IVI-3.10: Measurement and Stimulus Subsystems (IVI-MSS) Specification

March, 2008 Edition
Revision 1.0.1

Important Information

The IVI Measurement and Stimulus Subsystems Specification (IVI-3.10) is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at `www.ivifoundation.org`.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at `www.ivifoundation.org`.

Table of Contents

LIST OF FIGURES

IVI-3.10 Measurement and Stimulus Subsystems (IVI-MSS) Specification

IVI Driver Architecture Revision History

This section is an overview of the revision history of the IVI-3.10 specification.

Table 1. IVI-3.10 Revisions

Revision Number	Date of Revision	Revision Notes
Revision 0.1	July 24, 1999	Original draft. (Referred to as Section 12)
Revision 0.2	May 18, 2000	Removal of Asset Server details and introduction of new terminology
Revision 0.3	Nov 17, 2000	Removal of Event Server details in addition of UML diagram. This version was not complete in that the design rules and guidelines material had not been carried over from the prior version.
Revision 0.4	February 15, 2001	Complete reorganization of material.
Revision 0.5	April 21, 2001	Minor clarifications and inclusion of review comments.
Revision 0.6	May 15, 2001	New figures and numerous corrections and clarifications including material on the IVI configuration store.
Revision 0.7	Aug 20, 2001	Agilent copyright removed. Alphabetized Glossary. Several definitions reworded. Solution added. Clarifications and other minor changes as a result of review inputs.
Revision 0.8	September 10,2001	Brought style in-line with other IVI specs.
Revision 0.9	November 21, 2001	Added Requirements–Recommendations Section, Added XML Example
Revision 10.0	March 6-20, 2002	Results of telephone conferences held in March 2002
Revision 10.1	April 15, 2002	Acceptance of numerous format, style, and grammar errors.
Revision 10.2	April 18, 2002	Corrections to Naming Convention syntax and XML example
Revision 10.3	April 24, 2002	Review at Fort Collins IVI Meeting
Revision 10.4	July 20, 2002	Final Review draft for San Diego IVI Meeting
Revision 10.5	July 24, 2002	Final Review draft for IVI Foundation 2 week review
Revision 1.0	Oct 22, 2002	Approved by IVI technical Committee. Revision numbering reset.
Revision 1.0.1	October 22, 2007	Deprecate Event Server. Remove references to Event Server in sections 1.4, 1.6, 4.1, 4.2, 4.3, 4.4, 4.5.6, 5.2.1, 6.1 and 6.4.
Revision 1.0.1	March, 2008	Editorial change to update the IVI Foundation contact information in the Important Information section to remove obsolete address information and refer only to the IVI Foundation web site.

1. Overview of the IVI-MSS Specification

1.1 Introduction

The IVI Measurement and Stimulus Subsystem (IVI-MSS) Specification provides architectural guidance and design principles for software elements which utilize the IVI framework to provide specific software solutions abstracted one layer above the level of instrument-based control provided by IVI drivers.

The IVI-MSS specification contains architectural guidance and design principles needed to develop applications that provide two benefits beyond what is provided by IVI drivers alone. These are:

1. Providing for the creation of test and measurement solutions where multiple test assets are used together under a single software interface.
2. Delivering a higher degree of interchangeability through improved isolation of instruments from application software.

These two benefits come at the expense of additional cost and complexity relative to IVI drivers alone. Users who are willing to make some changes in their application programs when an instrument is interchanged or who do not need to aggregate several instruments together as a product or reusable solution that will interoperate with multiple vendors, will not need to follow this specification.

IVI-MSS is of particular benefit in the following applications:

1. Applications where the test system is short lived but which contain software elements that can be reused in future systems independent of instrumentation.
2. Applications that require instrument interchangeability in a way that minimizes the possibility of test program changes.
3. Applications that require the flexibility of interchanging instruments of different instrument classes.

The IVI-MSS specification provides a standard way to create measurement and stimulus abstractions that provide value beyond the capabilities of a physical test instrument and defines the use of the IVI Configuration Server to maintain portions of the system configuration.

1.2 Audience of Specification

The IVI-MSS specification is for software developers who provide test and measurement solutions. This specification refers to these developers as *solution providers*. A solution provider can be an instrument manufacturer, a third party, an end user, or a system integrator.

1.3 How this Document is Organized

Section 3, *IVI-MSS Architecture*, explains how IVI-MSS works and what its component pieces are. Section 4, *Component Model*, provides the details how the IVI-MSS components are used and how they work together in a COM-based system. Figures 3-1 and 3-2 are conceptual, whereas Figures 4-1 and 4-2 capture requirements that must be followed for a COM implementation of IVI-MSS. The details of C or other implementation technologies that may be used to implement the IVI-MSS principles are not contained in this specification.

Section 5, *Client Architectures*, reviews how client application programs use IVI-MSS solutions. Appendix A provides a set of design principles upon which IVI-MSS is based. Appendix B provides an example IVI-MSS solution that shows details of how the various IVI-MSS pieces are configured for use.

1.4 IVI-MSS Overview

A key aspect of this specification is the requirements of how several IVI components are to be utilized together in building a PC based test and measurement solution. The specification is an architectural framework that shows how the following IVI components are to be used:

- IVI Configuration Server

- IVI-COM Session Factory
- IVI Locking Component (future development planned)

This specification defines the following two additional types of components:

- IVI-MSS role control modules
- IVI-MSS servers

The definition of the semantic interfaces required for completed IVI-MSS solutions are outside of the scope of this specification. These interfaces are used by a client of an IVI-MSS Solution to access functionality. The definitions for these interfaces may come from any of the following:

- Existing IVI class specifications.
- Future IVI class specifications
- IVI Signal Interface Specification
- Customized interface definitions

Aligning these interfaces with existing standards may decrease the number of test program changes needed when replacing an IVI-MSS solution.

1.5 Relationship between IVI-MSS and IVI Drivers

IVI class-compliant specific drivers provide standardized API functions for test assets and by themselves provide a certain measure of instrument interchangeability. It is possible to achieve instrument interchangeability with IVI drivers if an application does not use instrument functions outside of the common set of IVI class-compliant specific driver functions and if the response to these functions from an alternative instrument meets the user's requirements. In such cases, the additional specifications of IVI-MSS are not needed to achieve instrument interchangeability.

Neither an instrument vendor nor a driver provider can guarantee that an interchanged asset will deliver the same answer to an application program if the only thing considered is the availability of an the IVI class driver interface. This is because the IVI class driver specifications do not rigidly define the actual instrument behaviors associated with the standardized function calls or methods. Thus, changes to the user application might be required when interchanging instruments using IVI drivers. By using IVI-MSS principles you isolate the code that might change. This makes it possible for a solution provider to guarantee that the target system will produce the "same answer" after an interchange.

1.6 References

The following documents are related to this specification.

- IVI-3.1: Driver Architecture Specification
- IVI-3.4: API Style Guide
- IVI-3.5: Configuration Server Specification
- IVI-3.6: COM Session Factory Specification
- IVI-3.11: Signal Interface Specification

1.7 Definitions of Terms and Acronyms

Term	**Definition**
hardware asset	Refer to *IVI-5.0: Glossary*.
Interchangeability	The ability to interchange an instrument of one type with that of another type, make, or model as long that the new device can perform the same physical operation. When performing this interchange using MSS, no change in the user application is required except replacing Role Control Modules.

IVI Configurable Component	Refer to *IVI-3.5: Configuration Server Specification*. IVI-MSS components are IVI Configurable Components.
IVI-MSS hardware asset interface	Refer to *IVI-5.0: Glossary*.
IviSoftwareModule	Refer to *IVI-5.0: Glossary*.
Measurement and Stimulus Subsystems (MSS)	Software architecture for building IVI-MSS solutions using this specification.
IVI-MSS server	Refer to *IVI-5.0: Glossary*.
IVI-MSS solution interface	Refer to *IVI-5.0: Glossary*.
IVI-MSS solution	Refer to *IVI-5.0: Glossary*.
Role	Refer to *IVI-MSS role* in *IVI-5.0: Glossary*.
Role control module (RCM)	Refer to *IVI-MSS role control module (RCM)* in *IVI-5.0: Glossary*.
SCPI	Standard Commands for Programmable Instruments. A standard for the semantics of alphanumeric commanded used to control instruments.
Second-order effects	Characteristics instruments that cause differences in functionality under the same interface function.
Solution	A test and measurement capability that includes instruments or other hardware devices and associated software to perform a useful function in a specific application area.
solution provider	The organization that develops, delivers, and supports a test and measurement solution.
TPS	Test Program Set. A TPS is a program that performs testing operations on an electronic module. It is directly associated with the requirements and capabilities of the module being tested.

2. When MSS is Required

The simplest application of IVI technology is when an application directly utilizes an instrument via an IVI class-compliant specific driver. In the following cases this simple model breaks down and additional steps must be taken to achieve desired results:

- The simple model does not provide the ability to use an instrument of one type to fulfill the functions served by an instrument of another type. There are cases when two dissimilar instruments are capable of performing the same physical operations. For example, an oscilloscope can make the same DC voltage measurement as a digital multimeter. Although IVI classes are defined for both of these instruments, the interface is entirely different and thus the instrument cannot be interchanged.
- The simple model does not address the case where differences in the measurement technology require slightly different set-up of two similar instruments. For instance, one instrument may require a delay between presenting an input and initiating a measurement. With the simple model, there are only two places where this instrument specific software can be put, the IVI driver or the application. Placing user code in a modified IVI driver has the undesirable ramification of changing the driver, which frequently comes from a 2nd party and for which the source code is often unavailable. Once a driver has been modified for use by an application, support for the driver from its provider may be lost. Placing user code in the end application complicates it by introducing set-up functions that are not normally in the domain of the end application. This can require that the application be fully tested and validated after the introduction of a new instrument. To address these issues, a new place needs to be identified where this code can be put, and this additional layer of code should be under control of whoever is responsible for achieving instrument interchangeability.
- The simple model that associates one driver with one hardware asset, does not provide a way to create virtual instruments or solutions that require more than one physical instrument but which present a single interface or API to an application program.
- The simple model does not provide a place to encapsulate measurement algorithms that are unrelated to a hardware asset. Frequently, these algorithms are sufficiently complex that it is desirable to encapsulate them into a reusable form. Achieving a high degree of reusability requires adding an abstraction level that is higher than general-purpose instrument drivers. A means is needed to achieve measurement or stimulus reuse that is independent of specific instrumentation.
- The simple model does not identify a specific software module that is controlled by an entity responsible for instrument interchangeability. Such a layer is necessary if some entity is required to maintain instrument independence. The driver level is not suitable because the instrument vendor will typically control it. Similarly, the higher-level software will concentrate on completing the measurement independently of instrument variations and will be owned by another entity. Therefore, an additional layer where the interchange can occur needs to be defined.

3. IVI-MSS Architecture

This section explains how IVI-MSS works and what its component pieces are.

3.1 Architectural Overview Example

This is an overview of an example IVI-MSS solution. In this simple example, three instruments are utilized together to deliver a solution to the user application. FW represents instrument firmware. HW represents instrument hardware.

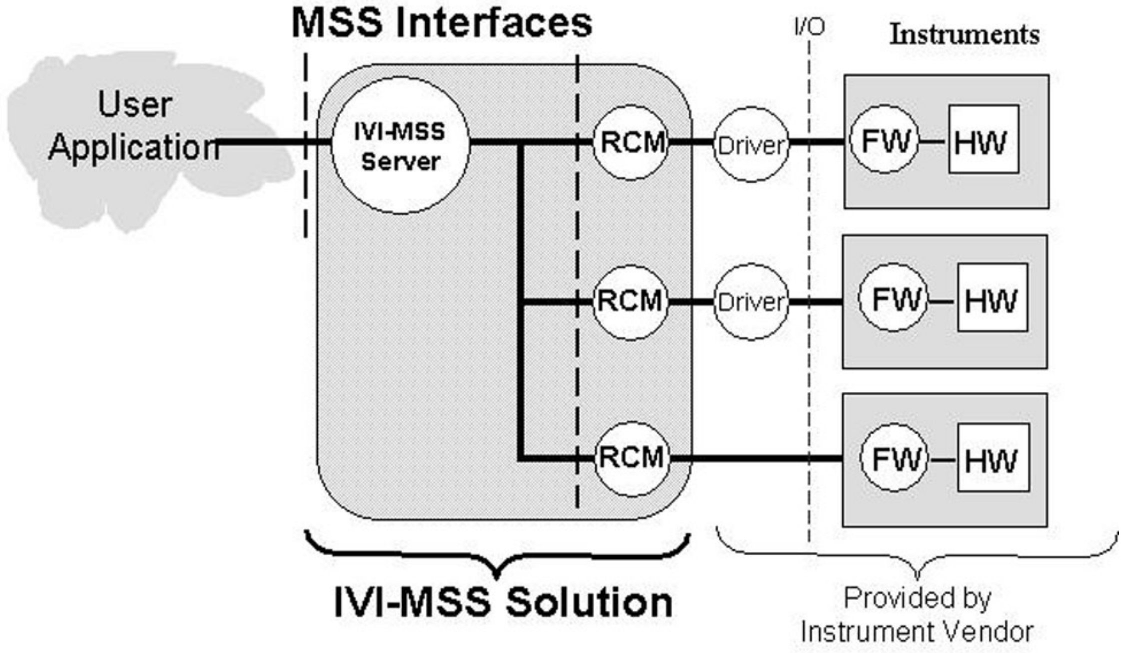

Figure 3-1 – Example IVI-MSS Solution

3.1.1 User Application - API Technology

The user application is the software environment in which an IVI-MSS Solution is utilized. Any application development environment can be used that is capable of interfacing with the software interface provided by an IVI-MSS Server.

The IVI-MSS specification provides architectural guidance and design principles for software components which utilize the IVI framework. Unlike IVI instrument drivers, which require standardized architectures and interfaces, IVI-MSS systems are often custom. As a result, the solution provider and the end user of the IVI-MSS system select the programming language and interface technology to use.

An IVI-MSS software component may export interfaces in the form of COM or C APIs similar to those used for standard IVI drivers, or it may use interface styles native to environments such as LabVIEW, MATLAB, or VEE.

In all cases, IVI-MSS solutions use the IVI Configuration Server to record and access all IVI-MSS configuration data. IVI-MSS components are recorded in the IVI configuration store using the IVI-MSS naming conventions set forth in Section 4.5, *Component Requirements*. The naming conventions make it possible to discover various IVI-MSS components when examining the IVI configuration store. The IVI Foundation defines both a COM API and a C API to the IVI Configuration Server. IVI-MSS systems access the IVI configuration store using one of these two APIs.

For IVI-MSS solutions that use COM interfaces, the IVI Foundation requires the use of the IVI-COM Session Factory for dynamic loading of components such as role control modules. Refer to *IVI-3.6: IVI-COM Session Factory Specification* for more information.

For IVI-MSS solutions that use C interfaces, the IVI Foundation requires the use of the C Shared Components for dynamic loading of components such as role control modules. Refer to *IVI-3.9: C Shared Components Specification* for more information.

For IVI-MSS solutions that use interfaces native to environments such as LabVIEW, MATLAB, or VEE, the solution provider determines the most appropriate mechanism to use for dynamic loading of components.

3.1.2 IVI-MSS Solution

The software modules that make up an IVI-MSS Solution encapsulate a specific test and measurement problem. Placing this type of software in a set of components that are independent of the user application provides for reuse of the encapsulated functionality.

3.1.3 IVI-MSS Solution Interface

Additional interface layers appear in an IVI-MSS solution beyond the simple model where a user application communicates directly with an instrument's IVI driver. IVI-MSS solution interfaces with COM or C APIs have the same style as IVI-COM or IVI-C drivers. What makes these interfaces unique are their semantics. Refer to *Section 3.2.2, IVI-COM and IVI-C API.*

3.1.3.1 *IVI-MSS Server*

The IVI-MSS server is a software module that encapsulates the functionality of a specific test and measurement domain independent of any specific instrumentation. An IVI-MSS server does not contain any code that is dependent on either the interfaces or behavior of any specific piece of instrumentation.

When the capabilities of an instrument impose limitations to what an IVI-MSS server can do, data that is descriptive of these limitations are provided to the IVI-MSS server in a device independent manner. For example, if an IVI-MSS server needs to know the maximum voltage a power supply provides, this information should be provided via the IVI-MSS role interface without disclosing to the IVI-MSS server the make and model of the power supply device. An IVI-MSS server does not directly communicate with test instrumentation or a general-purpose instrument driver. All test instrument interactions are done via IVI-MSS role control modules (RCMs).

Making good decisions on which code should be put in an IVI-MSS server and which code should be put in an RCM is critical to the design of an IVI-MSS solution. Any code that might need to be changed when an instrument is replaced should not be placed in an IVI-MSS server. IVI-MSS role control modules are the place to put code that is needed to adapt any specific instrument to the requirements of its associated IVI-MSS server.

An IVI-MSS server with a COM or C API is an IVI Configurable Component that conforms to the requirements in *IVI-3.1: Driver Architecture Specification*, *IVI-3.4 API Style Guide*, and *IVI-3.5: Configuration Server Specification*. Refer to Section 3.2.2*, IVI-COM and IVI-C API.*

3.1.3.2 *Role Control Module (RCM)*

An RCM is a software module that connects an instrument or instrument driver with an IVI-MSS Server. The API for an RCM is the key interface for interchangeability. Replacing an RCM along with its associated physical instrument is how an interchange is accomplished. An RCM should not expose asset peculiarities through its interface.

An IVI-MSS role control module with a COM or C API is an IVI Configurable Component that conforms to the requirements in *IVI-3.1: Driver Architecture Specification*, *IVI-3.4 API Style Guide*, and *IVI-3.5: Configuration Server Specification*. Refer to Section 3.2.2, *IVI-COM and IVI-C API.*

3.1.4 Instruments

In the example shown in Figure 3-1, *Example IVI-MSS Solution*, three instruments are required by the IVI-MSS solution. Two of these instruments are shown with IVI class-compliant specific drivers and one of them is shown as being used without an IVI driver.

3.2 Role Control Module (RCM) Requirements

The provider of an IVI-MSS solution shall supply an RCM for each test instrument or asset that is used. An RCM may also be used for specialized software modules that are hardware independent. An RCM implements an interface to be used by an IVI-MSS server. Any hardware asset specific software code that pertains to an IVI-MSS Solution shall be placed in RCMs.

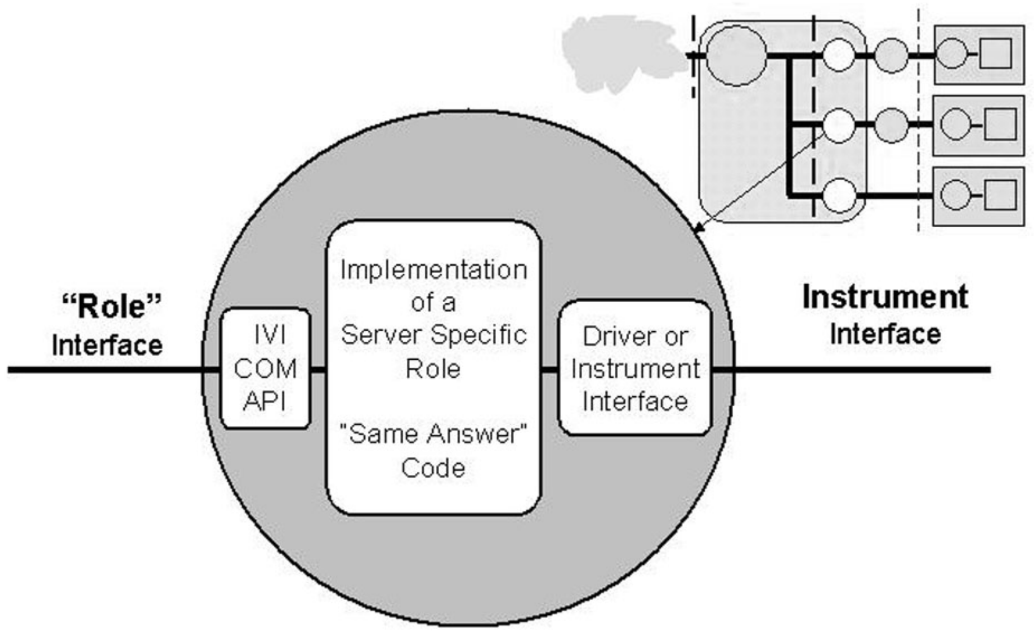
"Role"
Interface
IVI
COM
API
Implementation of a Server Specific Role
"Same Answer" Code
Driver or Instrument Interface
Instrument
Interface

Figure 3-2 – Inside a Role Control Module (RCM)

3.2.1 Role Interfaces

A role is an interface specification established between an IVI-MSS server and an RCM. It is defined by the solution provider and delivers the specific functionality that is needed by the associated IVI-MSS server. RCMs are typically solution-specific. Subsystems with different measurement requirements shall use different RCMs, even if utilizing the same asset. This is necessary to keep the burden of maintaining an instrument independent interface manageable. For example, a traditional test instrument may have hundreds of functions whereas an RCM may only have a dozen. If the user replaces this instrument without using an RCM, it may not be possible to know which of the functions of the original instrument are needed. In that case, the full set of functions may have to be provided and verified in the replacement driver or instrument.

The interface specifications for RCMs are the responsibility of their client or whoever it is that is making the claim of interchangeability. Whoever defined the RCM roles and the associated IVI-MSS Server is responsible for the development and verification of RCMs for that role.

In an IVI-MSS solution, the role interface is the key interface of interchangeability. The specific semantics defined for any given role shall not identify a particular instrument type so that it will be possible to interchange instruments of one type with that of another when the required capabilities are common.

A role interface shall not expose functionality beyond what is required by its associated server. This requirement increases the likelihood that alternative instruments can be found because it reduces the complexity of validating an RCM. Since role interfaces are the interface of interchangeability in an IVI-MSS solution, validating at this interface is necessary to "guarantee" the results after an interchange. The complexity of role interfaces determines the complexity of the validation task.

RCMs are defined and maintained by the solution provider. This means that the solution provider can guarantee that the solution produce the same or an acceptable result after an interchange. The solution provider accomplishes this by providing new RCMs and developing the "same answer code" that is contained therein. It is not possible to use an RCM to achieve interchangeability with an instrument that cannot perform the physical operations required by the associated role interface.

3.2.2 IVI-COM and IVI-C API

An IVI-MSS role control module with a COM or C API is an IVI Configurable Component that conforms to the requirements in *IVI-3.5: Configuration Server Specification.* This means that the IVI Configuration Server is used to configure an IVI-MSS solution so that IVI-MSS servers are logically connected to their associated set of role control modules.

IVI-MSS role control modules and IVI-MSS servers comply with selected requirements of *IVI-3.1: Driver Architecture Specification* and *IVI-3.4: API Style Guide*, as indicated here.

Since the *IVI-3.1: Driver Architecture Specification* is targeted towards IVI drivers not IVI-MSS components the following requirements are not applicable to MSS: specifying compliance, interchangeability checking, inherent capabilities, and some aspects of installation such as the IVI Standard Directory Tree. In particular there are no requirements on the installation location of IVI-MSS solution components. The other requirements pertain.

All of requirements in the *IVI-3.4: API Style Guide* pertain except for those specifically formulated for IVI Drivers an indicated in the guide.

3.2.3 RCM Naming

For the naming conventions for RCMs, refer to Section 4.5.4, *IVI-MSS Naming Conventions*.

3.2.4 Implementation of Server Specific Role

An RCM shall include all software code or algorithms that are necessary to adapt the capabilities of a specific test instrument or asset to the interface requirements of its role as specified by its associated IVI-MSS server. In some cases, where the requirements of a role are much simpler than the capabilities of a selected instrument, the RCM acts as a filter on the feature set of the more general-purpose instrument. In cases where the requirements of a role exceed the capabilities of a selected instrument, the RCM shall contain the software that is necessary to provide those capabilities (where possible).

3.2.5 Role Naming

For the naming conventions for roles, refer to Section 4.5.4, *IVI-MSS Naming Conventions*.

3.2.6 Driver or Instrument Interface

The RCM may do whatever is necessary to communicate with its associated hardware asset. An IVI driver can be used for asset communication but is not required. Using an IVI driver provides the benefit that in some cases, an interchange may be possible without creating a new RCM.

Test instrument or asset producers are responsible for providing the primary instrument control interface for their assets. Examples of these are SCPI interfaces, VXI*plug&play* driver*s*, or IVI drivers. When the test asset is a general-purpose instrument, it is not likely that the test instrument vendor would be responsible for developing or proving the RCMs needed by an IVI-MSS Solution.

4. Component Models

This section provides the details how the IVI-MSS components are used and how they work together in a COM-based system. Although the requirements specified in this section are focused on COM implementations of IVI-MSS, the general approach applies to IVI-MSS implementations that use other interface styles.

4.1 Simplified Model

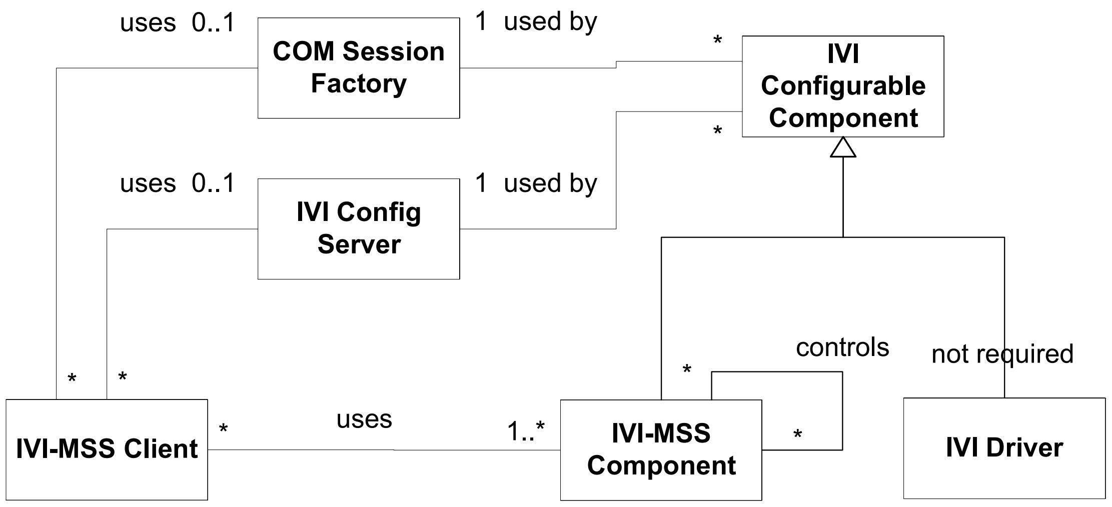

Figure 4-1 – IVI-MSS Simplified Component Model

Figure 4-1 is a simplified view that shows how the IVI-COM Session Factory and IVI Configuration Server components are used in conjunction with other software components to create an IVI-MSS solution. Both IVI-MSS servers and IVI-MSS role control modules are IVI-MSS components. An IVI-MSS solution can contain any number of these components, all of which are IVI Configurable Components. The IVI Configuration Server records the key configuration data and component names in the IVI configuration store. The IVI-COM Session Factory is used to instantiate all of the IVI-MSS components. IVI drivers are configured the same as IVI-MSS Components but they are not required.

4.2 Detailed Component Model

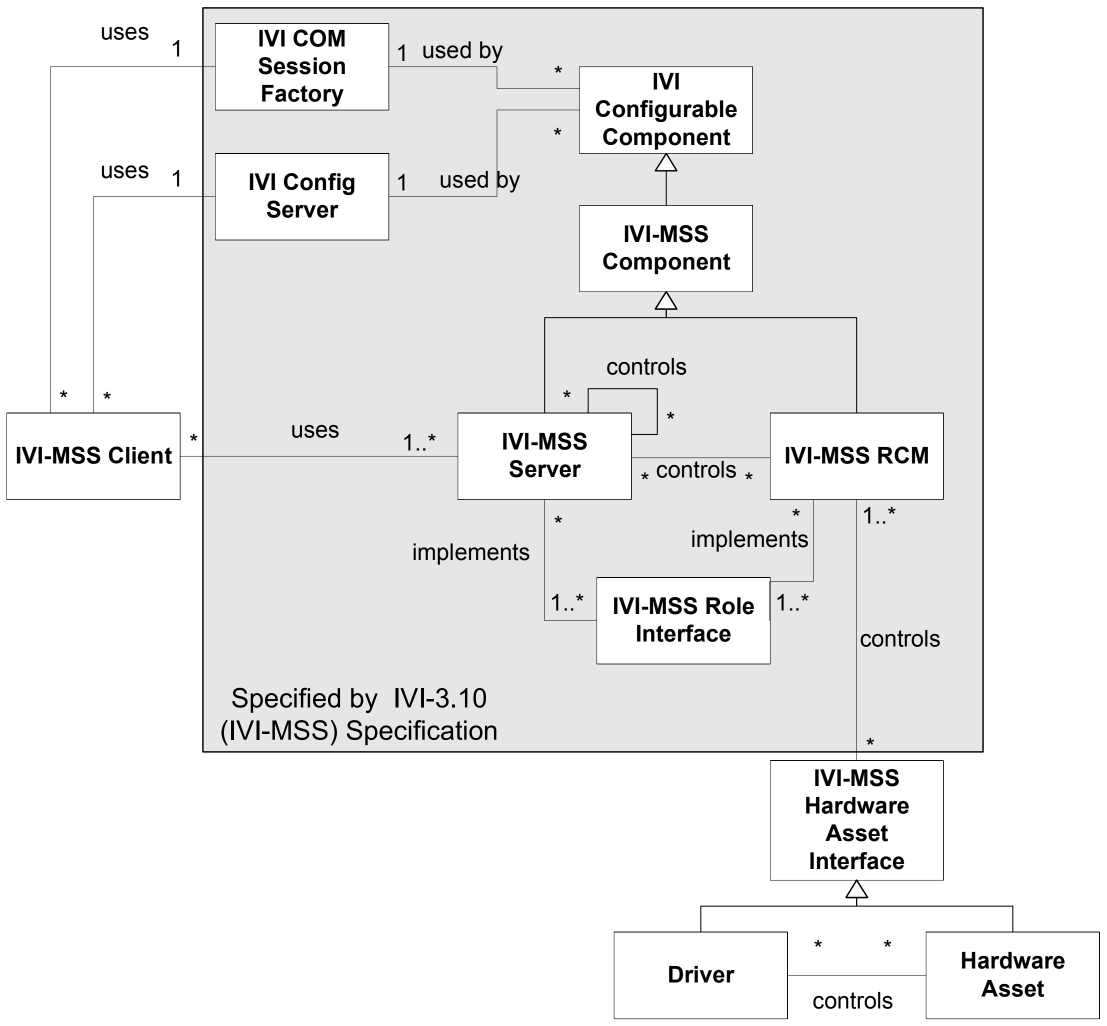

Figure 4-2 – IVI-MSS Full Component Model

The shaded areas in Figure 4-2 are covered by this specification. The full component model shows the additional detail of how IVI-MSS servers and IVI-MSS role control modules can be assembled together into collections to build a single solution that uses a set of hardware assets. The IVI-MSS client and IVI-MSS hardware asset interface are conceptual, and their content is not defined by this specification.

4.3 Actors

The actors are the set of software components that participate in the key use cases for IVI-MSS solutions.

- IVI-MSS Client
- IVI-COM Session Factory
- IVI Configuration Server
- IVI-MSS Servers

- IVI-MSS RCMs
- Role Interfaces
- IVI-MSS Hardware Asset Interfaces
- Hardware Assets
- Optional drivers

4.4 Component Model Walkthrough:

Starting at the outside pieces first, IVI-MSS hardware asset interfaces are controlled by one or more role control modules. The IVI-MSS hardware asset interface may be a hardware asset, or it may be a driver that operates a hardware asset. When a driver is utilized, it can be an IVI driver, VXI*plug&play* driver, or any other type instrument driver. The driver can be compiled into an associated RCM or called from the RCM and still fit the model. The second outside piece is the IVI-MSS client. This is any client application that communicates with an IVI-MSS server.

The common components inside the system are:
- The IVI-COM Session Factory, which is used to instantiate a particular class, RCM, or server.
- The IVI Configuration Server, which is used to store and recall the configuration information for a particular component. This configuration information could include such things as a bus address for a hardware resource. The IVI Configuration Server is also defined and provided by the IVI Foundation.

The final set of components is usually unique to each particular IVI-MSS system. The main piece is one or more IVI-MSS servers, which can control zero or more IVI-MSS servers and zero or more RCMs. Each IVI-MSS server can implement zero or more role interfaces that are usually, but not always, unique to that IVI-MSS system. A server that does not implement any role interfaces may not use a physical asset since all physical assets are to be accessed via RCMs. Each RCM can implement zero or more role interfaces and also control zero or more hardware assets.

The client interface of an IVI-MSS server is also a role interface. For IVI-MSS servers that are accessed directly by the user application, the role that is the primary interface from the IVI-MSS server to the user application is called the *IVI-MSS solution interface*.

A sample scenario would have the IVI-MSS client requesting a particular IVI-MSS server configuration from the IVI Configuration Server, and then using the IVI-COM Session Factory to instantiate that server. In turn that server would request the configurations for the RCMs that it needs and get those RCMs instantiated. The Client would then communicate to the server through a particular role interface and receive callbacks from the servers. After initialization, each RCM would configure its I/O route and start communicating with its particular hardware asset. This is just one scenario; many more could be generated using this diagram.

4.5 Component Requirements

4.5.1 IVI-MSS Client

One or more clients may connect to an IVI-MSS solution. Client applications can be stand alone graphical user interfaces or other applications that do not involve human interaction.

4.5.2 IVI Factory

There is a single out-of-process component called “IVI-COM Session Factory” per physical machine. Refer to *IVI-3.6: COM Session Factory* for more information.

4.5.3 IVI Configuration Store

The IVI configuration store is used to record all IVI-MSS configuration data. IVI-MSS components shall be recorded in the IVI configuration store using the naming conventions specified in Section 4.5.4, *IVI-MSS Naming Conventions*. The naming conventions make it possible to discover various IVI-MSS components when examining the IVI configuration store.

Role names are recorded in the IVI configuration store in a RoleNames collection, referenced from IviSoftwareModules and the IviConfigStore object. The collection is a string collection of role names as defined by the role name naming convention.

An IVI-MSS hardware asset interface is represented in the IVI configuration store by a combination of an IviHardwareAsset and an optional IviSoftwareModule.

Each IVI-MSS server and RCM is represented in the IVI configuration store by an IviSession object and its associated IVISoftwareModule. The IVI configuration store does not distinguish between RCMs and IVI-MSS Servers. Solution provider documentation shall specify which IVI-MSS roles are the primary entry points.

Refer to *IVI-3.5: Configuration Server Specification* for more information on the IVI configuration store.

4.5.4 IVI-MSS Naming Conventions

The following section describes naming conventions required for IVI-MSS components. This section also includes naming recommendations. Implementations that follow all of the recommendations should note this in the compliance document provided with implementation.

This section refers frequently to the example in Appendix B, *Example MSS Solution showing Configuration*.

4.5.4.1 Solution

A *solution* is the name for a particular IVI-MSS solution. In the example in Appendix B, the solution is "LevelFlight".

4.5.4.2 Role

A *role* represents a set of functionalities specific to an IVI-MSS solution. In the example in Appendix B, "Altimeter" is a role.

4.5.4.3 IVI-MSS Solution Server Name

Each IVI-MSS solution shall identify a unique IVI-MSS server for its intended client. This server shall be recorded in the IVI configuration store as an IviSoftwareModule and associated IviSessions. The IviSoftwareModule and associated IviSessions shall be identified in the IVI configuration store by an IviString data component of the following form:

IviDataComponent name := `MSSSolution`
IviDataComponent type := `string`
IviDataComponent value := <*solution*>

The recommended naming convention for the IviSoftwareModule for the unique IVI-MSS server is:

IviSoftwareModule name := `MSS` + <*solution*> + `Server`

In the example in Appendix B, the name IviSoftwareModule for the unique IVI-MSS server is `MSSLevelFlightServer`.

4.5.4.4 Role Names

Role names are used to form the names of the interfaces to IVI-MSS servers and role control modules. These are the formal names for an IVI-MSS role. Each role shall be recorded in the IVI configuration store as an IviPublishedAPI. The recommended naming convention in cases where the role is used in only one solution is:

IviPublishedAPI name := I`MSS` + <*solution*> + <*role*>

The required type for any IviPublishedAPI associated with IVI-MSS servers or role control modules is:

IviPublishedAPI type := `IVI-MSS`

In the example in Appendix B, the role name for the Altimeter is `IMSSLevelFlightAltimeter`.

4.5.4.5 Role Control Module (RCM) Names

RCM names are chosen by the solution provider. Each RCM shall be recorded in the IVI configuration store as an IviSoftwareModule and associated IviSessions. The recommended naming convention for the IviSoftwareModule in cases where the role is used in only one solution is:

IviSoftwareModule name := `MSS`+<*manufacturer*>+<*model number / family model number*>+<solution>+<role>

4.5.5 Discovery of IVI-MSS Components

A client application of the IVI Configuration Server can discover configured IVI-MSS solutions by using the following procedure:

1. Look for IviSessions and IviSoftwareModules that have attributes named `MSSSolution.` The value of these attributes will be the name of the solution that these IviSessions and IviSoftwareModules are the unique entry point for.
2. Follow associated links from the discovered IviSessions and IviSoftwareModules to identify all components of a particular solution.

4.5.6 Messaging Between Components

An MSS system may require asynchronous messaging between the components of an IVI-MSS solution or between an IVI-MSS Server and a client application. There are a variety of standard messaging techniques that can be used. Microsoft Message Queue (MSMQ) is the current recommendation. Each MSS system shall define conventions for specifying sources and destinations, so that messages can be correctly delivered, and so the destination component can identify the source. These conventions shall provide the ability to fully utilize the MSS system's abstractions (for example, allowing a role to be specified as a source or destination).

4.5.7 IVI-MSS Server

An IVI-MSS Server is an IVI Configurable Component that implements one or more role interfaces and controls zero or more RCMs. From a client perspective there is only one IVI-MSS server per IVI-MSS solution, and that server is the primary entry point for the solution. An IVI-MSS server cannot have any specific knowledge of any hardware asset.

An IVI-MSS server is represented in the IVI configuration store by an IviSession object and its associated IVISoftwareModule.

The server must utilize the IVI Configuration Server to identify the set of IVI-MSS role control modules to be used by the server. The IVI-COM Session Factory shall be used to acquire a handle to each RCM.

4.5.8 IVI-MSS Role Control Module (RCM)

There are zero of more RCMs per IVI-MSS solution. Zero is possible in the case where a server performs some type of algorithmic function that is not associated with hardware assets. The server must utilize the IVI-COM Session Factory to acquire a handle to its associated set of IVI-MSS role control modules. Each physical instrument in an IVI-MSS solution that is to be interchangeable shall have an associated RCM.

IVI-MSS role control modules implement one or more role interfaces and control zero or more IVI-MSS hardware assets Interfaces. An RCM shall not control other RCMs.

An RCM is represented in the IVI configuration store as an IviSoftwareModule. The configuration is represented in the IVI configuration store as an IviSesson, which references the underlying IviSoftwareModule implementation.

When a physical instrument is replaced in an IVI-MSS Solution, its associated RCM is replaced. If a physical instrument is replaced with two instruments that need to work together to fulfill the functions of the original instrument, two RCMs and an IVI-MSS server are required. The IVI-MSS server provides the role interface of the original RCM.

4.5.9 IVI-MSS Role Interface

The role interface of an IVI-MSS server encapsulates a set of attributes and functions needed by the IVI-MSS server to exercise a role in a particular IVI-MSS solution. The set of exposed attributes and functions should be as minimal as possible to maximize interchangeability. The role interface of a role control module is its API. Each RCM exposes one or more role interfaces as defined by the associated IVI-MSS server.

4.5.10 Distinction between RCMs and IVI-MSS Servers

Both RCMs and IVI-MSS servers are IVI Configurable Components IVI-MSS Components. They key difference is that IVI-MSS servers may utilize one or more RCMs or other IVI-MSS servers. RCMs may provide multiple role interfaces but they are not allowed to access other RCMs or IVI-MSS servers.

An IVI-MSS solution may include more than one IVI-MSS server. One of the IVI-MSS servers must be identified as the unique entry point to be used by client applications. This is accomplished through a unique naming convention in the IVI configuration store. Refer to Section 4.5.4.3, *IVI-MSS Solution Server Name* for more information.

4.5.11 IVI-MSS Hardware Asset Interface

An IVI-MSS hardware asset interface is the programming interface to a test and measurement instrument or other software controllable hardware device such as a card-based or modular instrument, which may or may not include an associated driver. The connections to a hardware asset's signal connectors are outside the scope of this specification.

5. Client Applications

5.1 Graphical User Interfaces

A complete ATE system may have several software layers. Each of these layers is a point where human interaction may be needed. Graphical user interfaces are helpful in developing and troubleshooting IVI-MSS solutions and in some cases are required by the end user. Specific layers where graphical user interfaces may be needed are the individual test instrument, IVI-MSS Role Control Module role interfaces, IVI-MSS Server interfaces, and the end user ATE interface. The details of graphical user interfaces are outside the scope of this specification.

5.2 ATE System Applications

5.2.1 Use of Multiple IVI-Servers

Multiple IVI-MSS solutions can be integrated together in a single ATE system. Each of these can provide unique measurement capabilities. Multiple instances of a single IVI-MSS solution can be utilized in cases where high throughput or parallelism is required. These can share from the same pool of physical test instruments or assets. When multiple IVI-MSS servers are used on the same physical computer, they share a single instantiation of the IVI Configuration Server. If the IVI Locking component is needed, a single instance is used by multiple IVI-MSS solutions.

5.2.2 Alternative Topologies involving Role Control Modules

In unique cases where a single test instrument or asset is used by multiple client applications and it is necessary to maintain asset state between the client applications, specialized IVI-MSS role control modules can be developed that expose multiple role interfaces that make it possible for a single software component to maintain the state of the associated physical device. This can be used to eliminate redundant device set up messages, which could improve performance.

5.2.3 ATE Specific Role Control Modules

In cases where an ATE (Automatic Test Equipment) system has a requirement for general purpose instrument functions that do not require the aggregation of instruments that would justify the overhead of an IVI-MSS server, the ATE system can be considered as taking over the role of the IVI-MSS server. An ATE system designed this way would communicate with its test assets via RCMs through an associated set of IVI-MSS role interfaces that are customized for the system.

In these cases a special set of IVI-MSS role control modules (RCMs) can be developed for this purpose. These RCMs must be owned, supported and maintained by the developers of the associated ATE system if claims of interchangeability are to be guaranteed.

6. Compliance

6.1 Solution Provider Responsibilities

A solution provider shall do the following to claim that a work product follows the IVI-MSS specification:

1) Provide a compliance document as outlined in Section 6.4*, Compliance Document Template.*

2) The logical names of all IVI-MSS servers and role control modules shall be placed in the IVI configuration store. The names shall be recorded by use of the IVI Configuration Server.

3) Both IVI-MSS Servers and RCMs shall be represented in the IVI Configuration Server by an IVISession and its associated IVISoftwareModule.

4) When the IVI-COM Session Factory is used in a solution, this shall be stated in the compliance document.

5) The IVI-MSS server that is the entry point for client interactions shall be named using the conventions specified in Section 4.5.4, *IVI-MSS Naming Conventions*.

6) The topology showing the hierarchy of IVI-MSS role control modules that are used to build a solution shall be provided.

7) The syntactic and semantic details of the role interface of IVI-MSS server that is the entry point for client access shall be documented.

8) The compliance document shall state whether the solution provider allows other parties to provide RCMs. When the solution provider allows other parties to provide RCMs the syntactic and semantic details of the role interfaces of all IVI-MSS servers and RCMs in the system shall be documented.

6.2 Verifying Interchangeability

Interchangeably is verified by maintaining a suite of tests that thoroughly exercises the interface of each IVI-MSS Role in the solution. The test suite is used before and after replacing an IVI-MSS Role Control Module and its associated physical asset. The results of this test are compared to determine if the interchanged asset is working properly and delivering acceptable results.

6.3 Customer Verification Process

The customer of an IVI-MSS solution can verify that the provider has complied with this specification by examining the IVI configuration store and verifying that all of the IVI-MSS components are visible for any asset that an interchange might be expected for. Specifically, any hardware device in the solution must have an associated IVI-MSS role control module where customization code can be placed if necessary. Where a solution is delivered with support for alternative devices for a given IVI-MSS role, compliance can be verified by switching the configuration data in the IVI configuration store. Performing this operation should cause the new device to be used without any other required interaction with the solution provider's product.

The customer should verify that all the required documentation as specified in Section 6.1, *Solution Provider Responsibilities*, is provided.

If the customer requires the ability to interchange devices without the services of the solution provider, the customer will need to obtain from the solution provider the interface details for any IVI-MSS role under which an interchange will take place. In addition, a test suite for this interface will also need to be obtained.

6.4 Compliance Document Template

The solution provider shall provide to users of an IVI-MSS solution a compliance document based on the following template.

Identification of IVI-MSS components

List the logical names of all IVI-MSS servers.

List the logical names of all IVI-MSS role control modules supported by the IVI-MSS solution. Refer to Section 4.5.4.5, *Role Control Module (RCM) Names*.

For each role control module indicate the make and model of the Hardware Asset it supports.

List the formal names of all roles in the solution. Refer to Section 4.5.4.4, *Role Names*.

Identify the name of the IVI-MSS server that is the entry point for client interactions.

Provide the topology or hierarchy of IVI-MSS role control modules that are used.

Identification of API style

State which of the IVI-MSS components use the IVI-C API style and which ones use the IVI-COM API style.

Disclosure of API Semantics

Document the syntactic and semantic details of the role interface of IVI-MSS server that is the entry point for client access.

State whether the solution provider allows other parties to provide RCMs. If so, provide the syntactic and semantic details of the role interfaces of all IVI-MSS servers and RCMs in the system that pertain to this agreement.

Utilization of the IVI Common Components:

State that the IVI Configuration store holds all of the configuration data for all IVI-MSS servers and role control modules in the solution.

State that the IVI Configuration server is used as the only means to record the names of IVI-MSS components into the IVI configuration store.

Appendix A: Design Principles

Following is a summary of the key concepts and design principles that pertain to IVI-MSS solutions. Adhering to these design principles is necessary to achieve the highest degrees of test asset interchangeability in ATE systems.

1. Measurements have value independent of instruments, applications, and Test Program Sets (TPS). A TPS is a program that performs testing operations on an electronic module. Encapsulating measurements into IVI-MSS solutions will reduce development costs by making them reusable.

2. A measurement's utilization of the features of instruments must be rigidly controlled by the use of the defined "roles" of IVI-MSS role control modules.

3. The use of IVI-MSS role control modules allows subsystem providers to guarantee a support life beyond the life of the utilized instruments.

4. To assure interchangeability it is critical to identify the owners of the various interfaces upon which interchangeability is dependent. The owners of these interfaces must be available and willing to support any claims of interchangeability. Support includes having the ability to test and verify capabilities and accuracies after the interchange.

5. A general-purpose full-featured interface of a test asset should not be the "interface of interchange" unless the client owns it and can make modifications. Client changes are often not possible with general-purpose instrumentation. Avoiding dependencies on the native full featured interface of a device is wise regardless of whether the interface is an ASCII or a driver interface. These interfaces must never be used directly by application programmers or TPS developers in systems that must guarantee interchangeability.

6. General-purpose interfaces that expose all of a test asset's features are difficult to test. For this reason, IVI-MSS role control modules that have tightly constrained interfaces make interchangeability testing easier and practical.

7. The ability to validate increases as API complexity decreases. When there is a small, well-defined interface that exposes the differences between two hardware assets, testing can be limited to this interface.

8. The ability to interchange test instruments increases as the ability to validate increases.

9. The abstraction of a measurement provides a natural isolation from asset peculiarities.

10. Extra layers of abstraction in a system naturally simplify the burden of test and make it easier to deliver interchangeability.
11. The number of potential interchangeable test assets is inversely proportional to the number of exposed features in an API.

Appendix B: Example MSS Solution showing Configuration

The following example shows the IVI configuration store details when two identical hypothetical MSS-based solutions are used in the same configuration environment. The IVI-MSS solution in this example includes a server and two associated RCMs. In the following example, there are three sets of hardware but only two sets are to be configured for use at any one time. This demonstrates how identical IVI-MSS solutions can be used in the same system. It also shows how a single IVI-MSS solution can have two sets of configured hardware available for its use though not at the same time.

The solution is named "LevelFlight". Its purpose is to measure altitude and make control adjustments to achieve level flight of an aircraft. The two roles used by this solution are "Altimeter" and "Elevator", and the roles names of the corresponding RCMs are `MSSLevelFlightAltimeter` and `MSSLevelFlightElevator`.

The example provides sufficient detail to see how the various software components are named and configured for use in a single computing environment using the IVI Configuration Server. Refer to *Figure 2-2:* IVI *Configuration Server UML Class Diagram*, in *IVI-3.5: Configuration Server Specification*.

Further details on the example can be found on the IVI Foundation web site. This includes an XML file that shows the configuration data of the example in the IVI configuration store along with a graphical model of the configuration data.

Description of configured components:

1. There are two LevelFlight solutions (subsystems) being used in the same computing environment. Both solutions use the same IVI-MSS server, but the solutions are configured to use different role control modules.
2. There are three sets of LevelFlight hardware available for use.
3. The names of the two solutions are chosen by the client as: `MyLevelFlightSolution1` and `MyLevelFlightSolution2`. These logical names are used to refer to IVI sessions representing associated IVI-MSS servers.
4. The solution provider determined that the primary entry point of the LevelFlight solution is through the `MSSLevelFlightServer` role.
5. Because there are two LevelFlight solutions, there are two IviSessions that refer to the IviSoftwareModule that is configured with the role name of `MSSLevelFlightServer`. The names of the two IVISessions are chosen by the client as `MyLevelFlightServer1` and `MyLevelFlightServer2`. Both IviSessions point to the single IviSoftwareModule configured by the solution provider and named `MSSLevelFlightServer`.
6. The IviSoftwareModule named `MSSLevelFlightServer` aggregates (controls) two RCMs. One RCM implements the `MSSLevelFlightAltimeter` role and the other implements the `MSSLevelFlightElevator` role.

   The client chooses the names of each RCM represented by an IviSession. For example, for `MyLevelFlightSolution1` the client names the two IVISessions that represent configured RCMs as `MyLevelFlightAltimeter1` and `MyLevelFlightElevator1`.

7. The following identifies the key actors referred to in the "Entry Owner" column of the table below:

   A1 is the person who writes the IVI-MSS client.
   A2 is the solution provider. The solution provider defines the interfaces for the top level server, any intermediate servers, and RCMs, and also provides the associated IVI-MSS servers and RCMs.
   A3 is the person who configures the system.

Three sets of hardware are available, each associated with an IviSession that represents an RCM. For example, three RCMs fulfill the `MSSLevelFlightElevator` role. The user has named the IVISessions for these RCMs as `MyLevelFlightEvelvator1`, `MyLevelFlightEvelvator2`, and

`MyLevelFlightEvelvator3`. Each IviSession refers to the IviSoftwareModule which implements the appropriate RCM that is supplied and named by the solution provider.

Term	IVI Configuration Server Entry	ConfigStore Type	Maintains Reference To	Entry Owner	Implements
Solution	MyLevelFlightSolution1	IviLogicalName	MyLevelFlightServer1	A1	
“	MyLevelFlightSolution2	IviLogicalName	MyLevelFlightServer2	A1	
Server Module	MSSLevelFlightServer	IviSoftwareModule	ElevatorRole AltimeterRole	A2	IMSSLevelFlightServer
Server Configuration	MyLevelFlightServer1	IviSession	MSSLevelFlightServer, MyLevelFlightElevator1, MyLevelFlightAltimeter1	A3 (A2 defines template)	
“	MyLevelFlightServer2	IviSession	MSSLevelFlightServer, MyLevelFlightElevator2, MyLevelFlightAltimeter2	A3 (A2 defines template)	
“	MyLevelFlightServer3	IviSession	MSSLevelFlightServer, MyLevelFlightElevator3, MyLevelFlightAltimeter3	A3 (A2 defines template)	
RCM	MSSManufModel1LevelFlightElevator	IviSoftwareModule		A2	IMSSLevelFlightElevator
RCM	MSSManufModel2LevelFlightElevator	IviSoftwareModule		A2	IMSSLevelFlightElevator
RCM	MSSManufModel3LevelFlightElevator	IviSoftwareModule		A2	IMSSLevelFlightElevator
RCM	MSSManufModel4LevelFlightAltimeter	IviSoftwareModule		A2	IMSSLevelFlightAltimeter
RCM	MSSManufModel5LevelFlightAltimeter	IviSoftwareModule		A2	IMSSLevelFlightAltimeter
RCM	MSSManufModel6LevelFlightAltimeter	IviSoftwareModule		A2	IMSSLevelFlightAltimeter
RCM Configuration	MyLevelFlightElevator1	IviSession	MSSManufModel1LevelFlightElevator, ManufModel12	A3 (A2 defines template)	
“	MyLevelFlightElevator2	IviSession	MSSManufModel2LevelFlightElevator, ManufModel13	A3 (A2 defines template)	

“	MyLevelFlightElevator3	IviSession	MSSManufModel3LevelFlightElevator, ManufModel14	A3 (A2 defines template)	
“	MyLevelFlightAltimeter1	IviSession	MSSManufModel4LevelFlightAltimeter, ManufModel15	A3 (A2 defines template)	
“	MyLevelFlightAltimeter2	IviSession	MSSManufMdel5LevelFlightAltimeter, ManufModel16	A3 (A2 defines template)	
“	MyLevelFlightAltimeter3	IviSession	MSSManufModel6LevelFlightAltimeter, ManufModel17	A3 (A2 defines template)	
Hardware Asset	ManufModel11	IviHardwareAsset		A3	
“	ManufModel12	IviHardwareAsset		A3	
“	ManufModel13	IviHardwareAsset		A3	
“	ManufModel14	IviHardwareAsset		A3	
“	ManufModel15	IviHardwareAsset		A3	
“	ManufModel16	IviHardwareAsset		A3	
Role / Interface	IMSSLevelFlightAltimeter	IviPublishedAPI		A2	
“	IMSSLevelFlightElevator	IviPublishedAPI		A2	
“	IMSSLevelFlightServer	IviPublishedAPI		A2	
Roles to aggregate	ElevatorRole	IviAPIReference	IMSSLevelFlightElevator	A2	
Roles to aggregate	AltimeterRole	IviAPIReference	IMSSLevelFlightAltimeter	A2	

Visual Basic program that will create the example’s configuration file

(Note the resultant XML file is available on the IVI Foundation web site.

```
Private Sub Form_Load()
Dim cs As New IviConfigStore
Dim ss As New IviSession

Dim MSSLevelFlightServer As New IviSoftwareModule
Dim MSSLevelFlightElevator1 As New IviSoftwareModule
Dim MSSLevelFlightElevator2 As New IviSoftwareModule
Dim MSSLevelFlightElevator3 As New IviSoftwareModule
Dim MSSLevelFlightAltimeter1 As New IviSoftwareModule
Dim MSSLevelFlightAltimeter2 As New IviSoftwareModule
Dim MSSLevelFlightAltimeter3 As New IviSoftwareModule

Dim ManufModel1 As New IviHardwareAsset
Dim ManufModel2 As New IviHardwareAsset
Dim ManufModel3 As New IviHardwareAsset
Dim ManufModel4 As New IviHardwareAsset
Dim ManufModel5 As New IviHardwareAsset
Dim ManufModel6 As New IviHardwareAsset

Dim MyLevelFlightServer1 As New IviSession
Dim MyLevelFlightServer2 As New IviSession
Dim MyLevelFlightServer3 As New IviSession

Dim MyLevelFlightElevator1 As New IviSession
Dim MyLevelFlightElevator2 As New IviSession
Dim MyLevelFlightElevator3 As New IviSession

Dim MyLevelFlightAltimeter1 As New IviSession
Dim MyLevelFlightAltimeter2 As New IviSession
Dim MyLevelFlightAltimeter3 As New IviSession

Dim MyLevelFlightSolution1 As New IviLogicalName
Dim MyLevelFlightSolution2 As New IviLogicalName

Dim APIRef As New IviAPIReference
Dim APIRef1 As New IviAPIReference

Dim pubSer As New IviPublishedAPI
Dim pubEle As New IviPublishedAPI
Dim pubAlt As New IviPublishedAPI

ManufModel1.Name = "ManufModel1"
ManufModel2.Name = "ManufModel2"
ManufModel3.Name = "ManufModel3"
ManufModel4.Name = "ManufModel4"
ManufModel5.Name = "ManufModel5"
ManufModel6.Name = "ManufModel6"

‘ add the hardware assets
Call cs.HardwareAssets.Add(ManufModel1)
Call cs.HardwareAssets.Add(ManufModel2)
Call cs.HardwareAssets.Add(ManufModel3)
Call cs.HardwareAssets.Add(ManufModel4)
Call cs.HardwareAssets.Add(ManufModel5)
Call cs.HardwareAssets.Add(ManufModel6)


MSSLevelFlightServer.Name = "MSSLevelFlightServer"
MSSLevelFlightAltimeter1.Name = "MSSManufModel4LevelFlightAltimeter"
MSSLevelFlightAltimeter2.Name = "MSSManufModel5LevelFlightAltimeter"
MSSLevelFlightAltimeter3.Name = "MSSManufModel6LevelFlightAltimeter"

MSSLevelFlightElevator1.Name = "MSSManufModel1LevelFlightElevator"
MSSLevelFlightElevator2.Name = "MSSManufModel2LevelFlightElevator"
MSSLevelFlightElevator3.Name = "MSSManufModel3LevelFlightElevator"

‘ add the software modules
Call cs.SoftwareModules.Add(MSSLevelFlightServer)
Call cs.SoftwareModules.Add(MSSLevelFlightElevator1)
Call cs.SoftwareModules.Add(MSSLevelFlightElevator2)
Call cs.SoftwareModules.Add(MSSLevelFlightElevator3)
Call cs.SoftwareModules.Add(MSSLevelFlightAltimeter1)
Call cs.SoftwareModules.Add(MSSLevelFlightAltimeter2)
Call cs.SoftwareModules.Add(MSSLevelFlightAltimeter3)

pubSer.MajorVersion = 1
pubSer.MinorVersion = 0
pubSer.Name = "IMSSLevelFlightServer"
pubSer.Type = "IVI-MSS"
pubEle.MajorVersion = 1
pubEle.MinorVersion = 0
pubEle.Name = "IMSSLevelFlightElevator"
pubEle.Type = "IVI-MSS"
pubAlt.Name = "IMSSLevelFlightAltimeter"
pubAlt.MajorVersion = 1
pubAlt.MinorVersion = 0
pubAlt.Type = "IVI-MSS"

‘ add the published APIs
Call cs.PublishedAPIs.Add(pubSer)
Call cs.PublishedAPIs.Add(pubEle)
Call cs.PublishedAPIs.Add(pubAlt)

APIRef.Name = "ElevatorRole"
APIRef1.Name = "AltimeterRole"
APIRef.UsedInSession = "Required"
APIRef1.UsedInSession = "Required"

Set APIRef.PublishedAPI = pubEle
Set APIRef1.PublishedAPI = pubAlt

‘ add the APIRef data components to the server
Call MSSLevelFlightServer.DataComponents.Add(APIRef)
Call MSSLevelFlightServer.DataComponents.Add(APIRef1)
Call MSSLevelFlightServer.PublishedAPIs.Add(pubSer)
Call MSSLevelFlightElevator1.PublishedAPIs.Add(pubEle)
Call MSSLevelFlightElevator2.PublishedAPIs.Add(pubEle)
Call MSSLevelFlightElevator3.PublishedAPIs.Add(pubEle)
Call MSSLevelFlightAltimeter1.PublishedAPIs.Add(pubAlt)
Call MSSLevelFlightAltimeter2.PublishedAPIs.Add(pubAlt)
Call MSSLevelFlightAltimeter3.PublishedAPIs.Add(pubAlt)

' add the MSS solution string data component
Dim solution As New IviString
solution.Name = "MSSSolution"
solution.ReadOnly = True
solution.UsedInSession = "Required"
solution.Value = "LevelFlight"
Call MSSLevelFlightServer.DataComponents.Add(solution)

MyLevelFlightServer1.Name = "MyLevelFlightServer1"
MyLevelFlightServer2.Name = "MyLevelFlightServer2"
MyLevelFlightServer3.Name = "MyLevelFlightServer3"
MyLevelFlightElevator1.Name = "MyLevelFlightElevator1"
MyLevelFlightElevator2.Name = "MyLevelFlightElevator2"
MyLevelFlightElevator3.Name = "MyLevelFlightElevator3"
MyLevelFlightAltimeter1.Name = "MyLevelFlightAltimeter1"
MyLevelFlightAltimeter2.Name = "MyLevelFlightAltimeter2"
```

```
MyLevelFlightAltimeter3.Name = "MyLevelFlightAltimeter3"

Set MyLevelFlightServer1.SoftwareModule = MSSLevelFlightServer
Set MyLevelFlightServer2.SoftwareModule = MSSLevelFlightServer
Set MyLevelFlightServer3.SoftwareModule = MSSLevelFlightServer

‘ configure the names of the required RCMs used by the servers
Dim elevatorRole As IviAPIReference
Dim altimeterRole As IviAPIReference
Set elevatorRole = MyLevelFlightServer1.DataComponents.Item("ElevatorRole")
Set altimeterRole = MyLevelFlightServer1.DataComponents.Item("AltimeterRole")
elevatorRole.Value = "MyLevelFlightElevator1"
altimeterRole.Value = "MyLevelFlightAltimeter1"

Set elevatorRole = MyLevelFlightServer2.DataComponents.Item("ElevatorRole")
Set altimeterRole = MyLevelFlightServer2.DataComponents.Item("AltimeterRole")
elevatorRole.Value = "MyLevelFlightElevator2"
altimeterRole.Value = "MyLevelFlightAltimeter2"

Set elevatorRole = MyLevelFlightServer3.DataComponents.Item("ElevatorRole")
Set altimeterRole = MyLevelFlightServer3.DataComponents.Item("AltimeterRole")
elevatorRole.Value = "MyLevelFlightElevator3"
altimeterRole.Value = "MyLevelFlightAltimeter3"

‘ set the session’s software module and hardware asset
Set MyLevelFlightElevator1.SoftwareModule = MSSLevelFlightElevator1
Set MyLevelFlightElevator1.HardwareAsset = ManufModel1
Set MyLevelFlightElevator2.SoftwareModule = MSSLevelFlightElevator2
Set MyLevelFlightElevator2.HardwareAsset = ManufModel2
Set MyLevelFlightElevator3.SoftwareModule = MSSLevelFlightElevator3
Set MyLevelFlightElevator3.HardwareAsset = ManufModel3
Set MyLevelFlightAltimeter1.SoftwareModule = MSSLevelFlightAltimeter1
Set MyLevelFlightAltimeter1.HardwareAsset = ManufModel4
Set MyLevelFlightAltimeter2.SoftwareModule = MSSLevelFlightAltimeter2
Set MyLevelFlightAltimeter2.HardwareAsset = ManufModel5
Set MyLevelFlightAltimeter3.SoftwareModule = MSSLevelFlightAltimeter3
Set MyLevelFlightAltimeter3.HardwareAsset = ManufModel6

Call cs.Sessions.Add(MyLevelFlightServer1)
Call cs.Sessions.Add(MyLevelFlightServer2)
Call cs.Sessions.Add(MyLevelFlightServer3)
Call cs.Sessions.Add(MyLevelFlightElevator1)
Call cs.Sessions.Add(MyLevelFlightElevator2)
Call cs.Sessions.Add(MyLevelFlightElevator3)
Call cs.Sessions.Add(MyLevelFlightAltimeter1)
Call cs.Sessions.Add(MyLevelFlightAltimeter2)
Call cs.Sessions.Add(MyLevelFlightAltimeter3)

MyLevelFlightSolution1.Name = "MyLevelFlightSolution1"
MyLevelFlightSolution2.Name = "MyLevelFlightSolution2"

Set MyLevelFlightSolution1.Session = MyLevelFlightServer1
Set MyLevelFlightSolution2.Session = MyLevelFlightServer2

Call cs.LogicalNames.Add(MyLevelFlightSolution1)
Call cs.LogicalNames.Add(MyLevelFlightSolution2)

Call cs.Serialize("c:\temp\MSS-Example.xml")
End Sub
```

Appendix C: Bibliography

- Architecture Drives Test Standards - Joe Mueller, Roger Oblad, IEEE Spectrum September 2000
- The Role of a Signal Interface in Supporting Instrument Interchangeability – Roger Oblad, Ion Neag, Autotestcon-2000 Proceedings, Piscataway, New Jersey. IEEE.
- Achieving Robust Interchangeability of Test Assets in ATE Systems - Roger Oblad, Autotestcon-1999 Proceedings, Piscataway, New Jersey. IEEE
- Connecting You to the Future - Ned Barnholt, Keynote Address Autotestcon-1998 Proceedings, Piscataway, New Jersey. IEEE.
- Applying New Software Technologies To Solve Key System Integration Issues - Roger Oblad, Autotestcon-1997 Proceedings, Piscataway, New Jersey. IEEE, pp.181-189.