# Security in Simulation – New Authorization Opportunities in HLA 4

*Björn Möller, Mikael Karlsson*
Pitch Technologies
Repslagaregatan 25
S-582 22 Linköping, Sweden
bjorn.moller@pitch.se, mikael.karlsson@pitch.se

*Reinhard Herzog*
Fraunhofer IOSB
Fraunhoferst. 1
76131 Karlsruhe, Germany
reinhard.herzog@iosb.fraunhofer.de

*Doug Wood*
MAK Technologies
13501 Ingenuity Dr, Suite 200
Orlando, FL 32826
dwood@mak.com

## Abstract

*A new version of the High Level Architecture (HLA) [1], with the working name HLA 4 [2], is under development. One security aspect that this new version addresses is authorization. The current version of HLA provides no standardized way to control which federates are allowed in a federation. HLA 4 adds new functionality for authorization, enabling control of which federates are allowed to connect to a Runtime Infrastructure (RTI) and to join a particular federation.*

*The new functionality is provided using a flexible plug-in design. Many types of authentication can be supported, for example passwords, X.509 certificates, and connection to Active/Open Directory servers. This paper describes a number of use cases for authentication in HLA federations. It also describes the technical design of the new security plug-in. Finally, it discusses the potential and limitations of the selected approach.*

## 1. Introduction

Distributed simulation is used in a wide range of domains, for example aerospace and defense, energy, medical, and transportation. In most of these, some level of security is required. For some applications, these requirements are the same as for any networked application. Common security measures such as encryption and firewalls are applied to protect systems and their transmitted data. For other applications, not the least in the defense domain, security requirements may be considerably more complex since systems, scenarios, and models may have higher or even different security classifications [3].

The fact that a given set of simulations has been federated to run a shared scenario indicates some level of trust, at least with respect to the information that is shared. This paper takes a closer look at how federates in a High Level Architecture (HLA) federation can be authorized to join a federation. Closely related to this is the ability for a federate to authenticate, i.e. to prove its identity to get authorization, which is also covered in this paper.

Note that there are a number other technical security measures like network and host access control, encryption, patching, malware detection, network monitoring, firewalls, logging, etc. that should be addressed when building a federation but that aren't covered in this paper.

### 1.1 Evolution of distributed simulation architectures

At the inception of distributed, federated simulation, data was shared by broadcasting all information from all the simulations across a local area network (LAN) [3]. Compared to contemporary federated simulations, these pioneering simulations were relatively small, dozens of simulations and a few hundred entities. So, frequent user datagram protocol (UDP)-based broadcast of complete entity state information was tractable.[1] These simulations focused on entity-level, real-time, defense applications, specifically training. Any simulation on the LAN could join and participate at any time.

The success of these simulations led to an inevitable expansion of requirements along all dimensions, i.e.:

- More entities and more simulations federated across larger, distributed networks
- Aggregate entity representation
- Faster and slower than real-time
- Non-defense applications including commercial, space, and civil
- Testing, analysis, and experimentation

The HLA was developed initially by the US Department of Defense (DoD) in response to these expanded requirements [5]. HLA added considerable functionality to address the performance impacts of growing federation size including the publication / subscription design pattern at both the object / interaction level and the object attribute level and data distribution management (DDM).

Another enhancement introduced by HLA is the concept of a defined set of simulations, referred to as federates, cooperatively participating in a federated simulation, referred to as a federation. As described in the HLA [1] standard:

> *"A named set of federate applications and a common federation object model (FOM) that are used as a whole to achieve some specific objective."*

This concept is part and parcel with the functionality of federation management, a group of HLA services for "creation, dynamic control, modification, and deletion of a federation execution".

This enhancement is significant from the perspective of the work presented in this paper because it defines the construct of a well-managed set of simulations designed to work in concert to meet a shared goal. This construct makes the HLA more suitable for the addition of mechanisms for authentication and authorization to control which federates may, and more importantly, may not join the federation.

### 1.2 Why authorization for HLA?

Early distributed simulation applications were often executed on dedicated local area networks (LANs). A simulation operating in the LAN was implicitly allowed to participate in the scenario. As simulation technology matured, was more widely used, and has moved out of closed labs, simulations needed to run on enterprise networks and wide-area networks (WAN) to reach their entire user base and fulfil the goal to be truly distributed.

From this evolution of being more widely distributed, a growing need was created for explicit authorization mechanisms in the HLA federation communication interface. One reason is that federations are now being developed and executed by multi-organizational and multi-national teams that need to communicate across enterprise networks. Other reasons are the increasing cloud deployment of federations including the provisioning of Modeling and Simulation as a Service (MSaaS) [6]. Strong authentication and authorization in an RTI also make federations more suitable for deployment in Zero Trust Network Architectures (ZTNAs) [7], an emerging security network approach, where network devices aren't trusted just because they are connected to a given enterprise network segment.

[1] Current DIS-based simulations are considerably larger.

### 1.3 Adding authorization to HLA 4

HLA is an open international standard through IEEE and maintained by SISO [1]. The HLA standard was developed using Balloted Products Development and Support Process (BPDSP) [8], which is itself a SISO standard. SISO standards are regularly revised, and a new version of HLA with the working name HLA 4 is currently under development. This activity is carried out by a SISO HLA Product Development Group (PDG) [2]. In short, the process consists of proposing new features or updates, collecting comments, discussing and drafting updates based on the comments, and finally balloting the new version of the standard.

In the first comment round for HLA 4, a member submitted a comment that it should be possible for a federate to provide a certificate when attempting to join a federation in order to prove that it has been tested and certified to comply with certain federation agreements [9]. After discussions within the group, it was agreed to adopt a slightly broader scope to the comment, one where the associated revision would also cover any kind of explicit authorization of federates. A Tiger Team was formed to develop a complete update to the specification, including a general description, a set of associated services, and APIs. As part of this, selected parts of the implementation were prototyped by two RTI implementers. A complete specification was included in Draft 2 of HLA 4, which is now available.

## 2. HLA 4 Authorization – external view

There are three important terms that the reader should understand to fully benefit from this section:

1. Authorization, which is the right to perform certain operations on a certain resource
2. Authentication, which is proof of identity
3. Credentials, which is information that proves that someone has certain rights, e.g. a password or certificate
4. Policy, which is the set of rules by which authorization is granted. This could be based on authenticated identity or other parameters.

### 2.1 Technical overview

In the simplest case, the new authorization mechanism in HLA 4 is used to determine whether a given federate is allowed to connect to an RTI, based on its credentials, as shown in Figure 1.

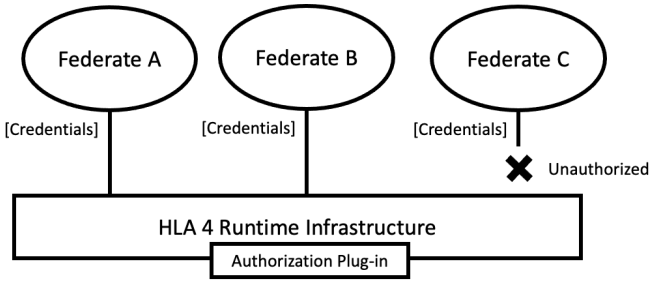


*Figure 1: Basic HLA 4 Authorization Use Case*

Each federate provides credentials when connecting to the RTI. The RTI uses an authorization plug-in to determine if a federate is allowed to connect. If a federate isn't allowed to connect, an exception is thrown. In this case, federate A and B provide valid credentials for connecting but not federate C.

More advanced use cases are also supported, such as federates only being allowed to connect to the RTI and join a given federation with a given name. If no authentication has been configured in the RTI, the default behavior is the same as in earlier HLA versions, i.e., any federate is allowed to join. All RTIs are required to provide a default simple password plug-in. More advanced plug-ins may be substituted for the default by the federation developers or third parties.

### 2.2 Sample use cases

Before describing the technical details, we would like to offer a number of intended practical use cases will be covered. The first three are shown in Figure 2.

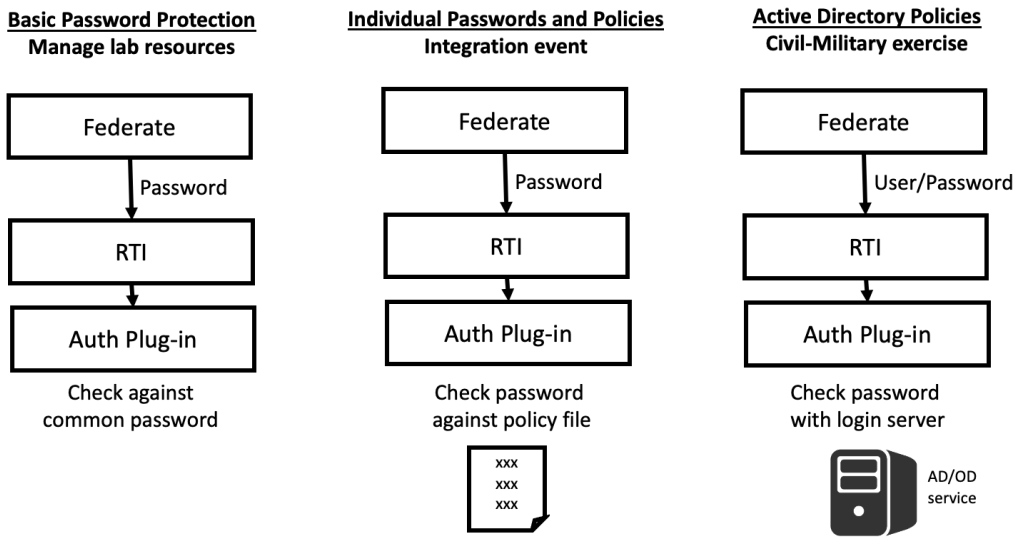


*Figure 2: Three Use Cases*

### 2.3 Use case: Basic password protection for managing lab resources

An organization has a simulation lab where a number of simulators are developed, and integration tested. A number of RTIs are available on the network for each team and for joint testing. It is considered critical that each team use only their assigned RTI and that nobody mistakenly uses an RTI currently assigned to other tasks.

As a solution, the system administrator specifies a different password for each RTI, which is encrypted and stored locally with each respective RTI (see the first column in Figure 2: Three Use Cases). These passwords are provided to each of the different development and integration teams based on their needs.

### 2.4 Use case: Individual passwords and policies during integration

Several organizations provide federates for a joint training exercise. They first need to perform integration testing in a dedicated network. It is considered critical to manage exactly which federate joins which federation, using a given federate name.

As a solution, the systems administrator can specify these requirements in a policy file for each RTI (see the second column in Figure 2: Three Use Cases). Each federate is assigned a different password. An example of a policy file for this is provided in Figure 3.

```
<RTIpolicy name="My sample policy">
   <federation name="AirBaseTraining23">
      <allowedFederate name="F16sim" password="GYtY73"/>
      <allowedFederate name="Visualizer" password="yGEo7T"/>
   </federation>
</RTIpolicy>
```

*Figure 3: Sample Policy File*

This policy specifies which federation executions are allowed (see the second line of the file), and for each federation execution, which federate names and which corresponding password are allowed (see lines three and four). Note that the above policy file format and content are for example only.

### 2.5 Use case: Active Directory policies in a Civil-Military exercise

A large crisis response exercise is performed over the course of a few weeks. Participants include defense organizations from several nations, police, NGOs and others. A large IT network is operated to support the exercise with mail, chat, file sharing, and other traditional IT services. User management, including policies, is handled using Active Directory (AD) or Open Directory (OD). There is a requirement to control which people are allowed to operate which simulator in which federation.

As a solution, the systems administrator can implement a plug-in that connects the RTI to the AD/OD server (see the third column in Figure 2: Three Use Cases). Policies are then specified in the AD/OD server. When a user logs into a training application, they provide their authentication details to the application, and the associated federate passes the authentication details on to the RTI. The RTI subsequently calls the AD/OD servers to authorize the user.

### 2.6 Use case: Certificates for proof of compliance

The three authorization approaches described above (sections 2.3 through 2.5) are mainly focused on making sure the joining federate has the appropriate credentials. However, the new HLA authorization concept allows more than providing a credential to prove the federate's identity. The authorization token can be anything. It can be a structured document describing the federate's identity and capabilities, and it can be certified and digitally signed by a trusted authority. That is called a digital certificate.

The motivation for such a certificate derives from the fact that a federated simulation environment represents a virtual environment used for a very specific purpose. For assuring trustiness, each federation participant has a specific role and must comply with a priori federation agreements. If a federate does not comply with these agreements, either because of insufficient code quality or simply because of some deployment issues, this federate may cause problems to the purpose and integrity of the federation. With a certificate, the federate can give a detailed and computer-readable description of its capability to comply with interoperability requirements identified by the federation agreements. And, most importantly, the RTI can deny access for unqualified federates.

The NATO Modelling and Simulation Group (NMSG) within the Science and Technology Organization (STO) is currently working on the installment of a certification process for federated simulations. Key to that will be the formulation of interoperability requirements (IRs). These IRs can be seen as atomic behavior elements, which are composed into behavior patterns called interoperability badges. Federate developers will refer to these badges when presenting their compliance statement. The NMSG is developing an infrastructure, including a tool set, to certify these compliance statements and to create a digital certificate signed by an accredited certification entity.

With the new HLA authorization feature, these certificates can now be used as a proof of compliance for joining federates. Thanks to the plug-in mechanism, anyone can provide an implementation of the authorization service. The NMSG is planning to use X.509 certificates[2] as defined in the ITU-T or the respective ISO/IEC 9594-8 standard. In addition to the usual public key validity check, the authorization service can provide further federation-specific configuration options like the interoperability requirements for each individual federate. With that, the authorization service is not just able to test if a joining federate is authentic but also if it complies with the required interoperability pattern and finally whether it owns a valid certificate from a trusted entity.

[2] See https://tools.ietf.org/html/rfc6818

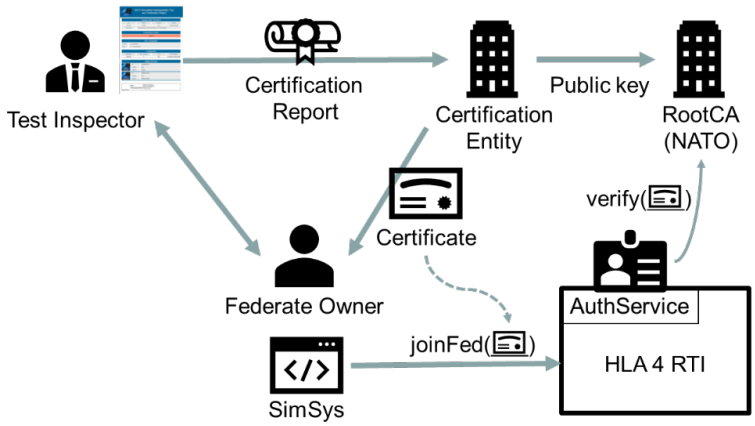


*Figure 4: NATO Approach to Interoperability Certification*

Figure 4 shows a workflow sketch of how the NMSG envisions testing simulation systems and providing certificates for the authorization service.

## 3. HLA 4 Authorization – Technical View

This section gives a view of the time technical architecture, the authorization interfaces and the datatypes. The new authorization mechanism consists of three main updates to the HLA standard, as shown in Figure 5.

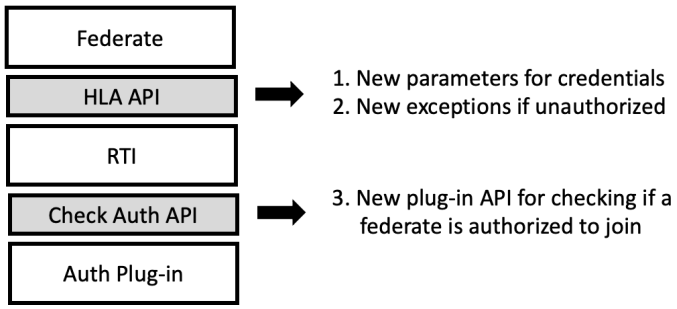


*Figure 5: Updates to the HLA Standard for Authentication*

The updates are as follows:

1. A federate can provide credentials when connecting to an RTI. This is done using a new parameter in the Connect service.
2. Where a federate is not allowed to connect to the RTI, create a federation execution, or join a federation execution, a number of new exceptions have been added to these HLA services.
3. The RTI determines if a federate is authorized to connect to the RTI, create a given federation, and join a given federation with a given name, based on the credentials provided in the Connect call. To do this, the RTI calls a plug-in using a standardized API.

The two main API components are the Credentials class and the Authorizer interface. Both of these are provided in C++ and Java in the standard with similar structure. Note that the **`Courier bold`** font is used for classes and interfaces in the text below.

### 3.1 Credentials and Authorizer

The RTI uses a federate's credentials and an authorization service to verify if a service invocation is permitted. The credentials and authorization service are represented in the programming language APIs by a **`Credentials`** class, an **`Authorizer`** interface, and an **`AuthorizerFactory`** interface.

**Credentials** is a concrete class that provides a simple envelope to contain the credentials type and the credentials data. An instance of the **Credentials** is provided to the RTI by the federate in the *Connect* service invocation. **Authorizer** implementations may provide **Credentials** subclasses with more sophisticated implementations and methods for constructing the **Credentials** instances. The **Credentials** class has a method getType() that returns the string name of the credentials type, for example "HLAplainTextPassword." A second method, getData(), returns the credentials data encoded using a method specific to the credentials type. The Java version of this class is shown in Figure 6.

```
public class Credentials {
   private final String _type;
   private final byte[] _data;
   public Credentials(String type, byte[] data) {
      _type = type;
      _data = data;
   }
   public String getType() { // Returns the type of these credentials.
      return _type;
   }
   public byte[] getData() { // Returns the encoded data of these credentials.
      return _data;
   }
}
```

*Figure 6: Java Credentials API*

The RTI interacts with the authorization service through an **Authorizer** interface. The **Authorizer** interface specifies methods to determine if a service invocation (*Connect*, *Create Federation Execution*, *Destroy Federation Execution*, and/or *Join Federation Execution*) is allowed for the federate, given its credentials. An implementation of the **Authorizer** interface may support more than one type of credentials. The RTI accepts or rejects the invocation of the service call based on the results of providing the credentials and other associated parameters supplied by the federate during the connect call to the authorizer. The implementation of an authorization service provides a concrete **Authorizer** class implementation derived from this interface. The Java version of this class is shown in Figure 7.

```
public interface Authorizer {
   // obtain the string name of the authorization service
   String getName();
   // authorize Connect given the federate's credentials
   boolean authorizeRtiOperation(Credentials credentials);
   // authorize Create/Destroy Federation Execution for the given federation name
   boolean authorizeFederationOperation(Credentials credentials, String federationName);
   // authorize Join Federation Execution for the given federation and federate name
   boolean authorizeFederateOperation(Credentials credentials, String federationName,
                                      String federateName);
}
```

*Figure 7: Java Authorizer API*

The RTI uses an **AuthorizerFactory** interface to create an **Authorizer** instance of a specific authorization service. The implementation of an authorization service provides a concrete **AuthorizerFactory** class implementation derived from this interface. The Java version of this class is shown in Figure 8.

```
public interface AuthorizerFactory {
   // obtain the string name of the authorization service
   String getName();
   // obtain an instance of the Authorizer implementation for the authorization service
   Authorizer getAuthorizer();
}
```

*Figure 8: Java AuthorizerFactory API*

Whether an RTI performs authorization or not and the specific authorization service the RTI uses are configurable through the RTI's initialization data (RID). A given RTI execution can be configured with only one authorization service.

### 3.2 Passing credentials to the RTI

A federate encodes the credential data itself and uses it to instantiate a **`Credentials`** instance. As stated previously, a specific authorization service may provide a **`Credentials`** subclass that makes this encapsulation of the credential data easier. The federate invokes the Connect service call passing the credentials to the RTI Ambassador. During connect, the RTI will pass the credentials to the authorization service via an **`Authorizer`** instance it created using the **`AuthorizerFactory`**. The authorizer will either accept or reject the provided credentials, resulting in a successful service call or an exception. Including the type name in the credentials enables the RTI and authorizer to verify that the federate has provided the correct type of credentials.

Typically, the authorizer will be maintained in a central RTI component (CRC), and the credentials will be passed to it. There is no requirement for the federate's local RTI component (LRC) to use or link with the authorization service package or library since it need only use the **`Credentials`** class included in the primary RTI package or library. It should be noted that the security of any communication of the credentials between networked RTI components is provided by the RTI implementation, e.g. TLS or HTTPS, and is outside the scope of the authorization service.

### 3.3 Providing a plug-in

All RTIs will provide an implementation of the HLA predefined authorization service named "HLAauthorizer." This authorization service supports the simple, predefined credentials type "HLAplainTextPassword." The implementation includes the following:

- A subclass of the **`Credentials`** class named **`HLAplainTextPassword`** with a constructor that takes a string and constructs an instance with credential type "HLAplainTextPassword" and credential data containing the string encoded as an HLAunicodeString.
- An implementation of the **`Authorizer`** interface called **`HLAauthorizer`**. This implementation supports the authorization of credentials of type "HLAplainTextPassword." It may support other credential types.
- An implementation of the interface **`AuthorizerFactory`** called **`HLAauthorizerFactory`**. This implementation creates instances of **`HLAauthorizer`**.

In addition to supporting this predefined authorization service, a factory-factory design pattern is used to support user-supplied authorization services similar to how HLA now supports user-supplied time implementations. To acquire an **`AuthorizerFactory`** implementation, the RTI will use an **`AuthorizerFactoryFactory`** class provided as part of the standard. The Java version of this class is shown in
Figure 9. The factory-factory class provides a method to get a specific named authorization service, e.g.,
`AuthorizerFactory authorizerFactory = AuthorizerFactoryFactory.getAuthorizerFactory("HLAauthorizer");.`

```
public class AuthorizerFactoryFactory {
   public static AuthorizerFactory getAuthorizerFactory(String authorizerName)
   {
      ServiceLoader<AuthorizerFactory> loader = ServiceLoader.load(AuthorizerFactory.class);
      for (AuthorizerFactory authorizerFactory : loader) {
         if (authorizerFactory.getName().equals(authorizerName)) {
            return authorizerFactory;
         }
      }
      return null;
   }
}
```

*Figure 9: Java AuthorizerFactoryFactory API*

## 4. Discussion

When it comes to security, there is always the question of how secure a system has to be. Raising security means raising costs. So, it is a good thing the HLA authorization approach is highly scalable from zero, in cases where no control over joining federates is required, up to the level suggested by a risk analysis, such as using a digitally signed certificate or proving the identity of a federate or its compliance with the current federation agreement. Solutions can vary to meet requirements. It might be as simple as a signed document, indicating the federates capabilities. Or the certificate can contain a digital fingerprint of the assigned federate, which may be used to verify authenticity and prevent accidental or even unauthorized reuse of certificates.

The specification titled "Internet Engineering Task Force Request for Comments 5755[3] [RFC 5755] explains the usage of the public-key concept to create a Public Key Certificate (PKC) and also Attribute Certificates (AC), which are often confused. The PKC can be considered like a passport: identifying the holder, tending to last for a long time, and hard to forge. The AC, on the other hand, is more like an entry visa: issued by an approved authority and possibly not lasting for as long a time. For the purpose of the HLA authorization certificate, an AC is most suitable for a role-based access control concept. The HLA authorization service may still need to ensure the joining federate is the rightful owner of the AC, so a reference to the PKC inside the AC might be required.

For authorization to be secure, it is important that the RTI use secure transportation for credentials between different components, e.g. the local RTI component (LRC) and the central RTI component (CRC). Otherwise, credentials can be eavesdropped. One example of such a transport mechanism is Transport Layer Security (TLS) 1.3 [10]. Exchanging hashed credentials openly on the wire won't add security since an eavesdropping application could record the hashed credentials and use it to establish an unwanted session with the RTI. Secure transportation will prevent such eavesdropping.

## 5. Conclusions

The new HLA Authorization service is a non-breaking change to the standard. Existing federations are not forced to change if they don't require authorization. However, the service enables an authorization capability, which might be required for trusted federations. And, with the growing application of federated simulation in areas where decisions depend on the integrity of federations or training must happen in valid virtual environments, trust is essential. But even better than trust is control. And the HLA authorization service is enabling just the level of control to the federation owners need by implementing authentication and authorization via federation-defined plug-ins.

## 6. Acknowledgements

The authors appreciate the technical contributions of Mr. David Drake and Dr. Katherine L. Morse to the development of this solution within the HLA PDG and their review of this paper.

## 7. References

[1] IEEE, "High Level Architecture for Modeling and Simulation," IEEE 1516-2010, 18 August 2010
[2] SISO, "Product Nomination for High Level Architecture, version 3.0", SISO-PN-016-2016, 11 January 2016
[3] NATO MSG-080 "Security in Collective Mission Simulation" STO-TR-MSG-080. 2013
[4] IEEE, "Distributed Interactive Simulation – Application Protocols," IEEE 1278.2012, 19 December 2012
[5] US DoD, "High Level Architecture for Modeling and Simulation," version 1.3, 20 April 1998
[6] NATO MSG-164, "Modeling and Simulation as a Service (MSaas) Phase 2"
[7] John Kindervag, "Webinar: Zero Trust Network Architecture", 2010, Forrester, www.forrester.com
[8] SISO, "Balloted Products Development and Support Process (BPDSP)," SISO-ADM-003-2011, 14 November 2011
[9] SISO, "Federation Engineering Agreements Template," 2 August 2013, SISO-STD-012-2013
[10] IETF, "Transport Layer Security (TLS) Protocol," version 1.3, ISSN," August 2018

[3] See https://tools.ietf.org/html/rfc5755

## Author Biographies

**BJÖRN MÖLLER** is the President and co-founder of Pitch Technologies. He has more than twenty-five years of experience in high-tech R&D companies, with an international profile in areas such as modeling and simulation, artificial intelligence and web-based collaboration. Björn Möller holds a M.Sc. in Computer Science and Technology after studies at Linköping University, Sweden, and Imperial College, London. He is currently serving as the chairman of the SISO RPR FOM Product Development group, chairman of the SISO Space FOM Product Development Group. and the vice chairman of the SISO HLA Evolved Product Development Group.

**MIKAEL KARLSSON** is the Infrastructure Chief Architect at Pitch overseeing the world's first certified HLA IEEE 1516 RTI as well as the first certified commercial RTI for HLA 1.3. He has more than ten years of experience of developing simulation infrastructures based on HLA as well as earlier standards. He also serves on several HLA standards and working groups. He studied Computer Science at Linköping University, Sweden.

**REINHARD HERZOG** is leading the research group "Modelling and System Networking" at the Fraunhofer IOSB. He is responsible for the development of conformance test systems for various communication protocols and the design of communication infrastructures and integration middleware systems, like the certified open source HLA run-time infrastructure GERTICO. Between 2003 and 2008 he has been giving lectures on "Simulation Technologies" at the University of Cooperative Education in Karlsruhe. Reinhard Herzog holds a M.Sc. in Computer Science (Informatik) after studies at the Karlsuhe Institute of Technology, Germany. Since 2012 he has been member of several NATO Modelling and Simulation Groups, dedicated to HLA interoperability.

**DOUG WOOD** has over thirty years of experience developing software for distributed simulation and training. He has performed and managed research and software development in distributed simulation architectures and protocols, automated behavior, electronic warfare simulation, emergency management simulation, and training device databases. He has served as a principal software engineer at MAK Technologies for the past 20 years working on the Link family of products involved in facilitating distributed simulation interoperability.