[white paper]

eicon
technology
corporation

OHIO

Understanding the Web-to-Host Application Programming Interface

November 1999

Jacinthe Paradis
Senior Program Manager, SNA Access Business Unit

Eugène Aresteanu
Senior Architect ,SNA Access Business Unit

EICON TECHNOLOGY

Table of contents

WHAT IS OHIO? 2

STATEMENT OF INTENT 2

EVOLUTION OF OHIO 2

OVERVIEW OF THE OHIO API 3

EICON TECHNOLOGY'S IMPLEMENTATION OF OHIO 4

Overview of Aviva for Java 4

Objects Relationships 6

The Relationship Between OHIO and Eicon's Aviva for Java Applet 8

Implementation Issues 9
Connect and Disconnect 9
Configuration 10
Send AID 10
Blocking Calls 10
Fields Collection Semantics 11
Error Handling 11

Backward Compatibility 11

CONCLUSION 12

References 12

What is OHIO?

OHIO simply stands for **O**pen **H**ost **I**nterface **O**bjects. Essentially, it is an API in-the-making, conceived by the industry's leading vendors of host access solutions, in answer to the growing requirement for a standard, vendor-neutral interface for Web-to-host applications. An off-shoot of the Internet Engineering Task Force (IETF), the OHIO working group put forth an initial draft proposal in September of 1998, which was well received.

While HLLAPI has been for many years the de-facto standard for writing applications that access host data, OHIO is a set of object classes that defines a new open programming interface for accessing host data, especially via the Internet. The first draft of OHIO deals with 3270 and 5250 display protocols but lays the foundation for growth toward other emulation types (VT, Bull, etc.). As corporations gradually migrate their host access applications to the Web, they are reluctant to adopt any one proprietary solution. To replace the traditional HLLAPI API, they need a new standard API (**A**pplication **P**rogramming **I**nterface) that can be adapted a) to the Web environment, and b) to technologies such as Corba, Java, ActiveX, while remaining compatible with implementations of 3270 and 5250 protocols. By developing applications using a standard API as opposed to one that is proprietary, corporations can protect their investments.

The intent of this white paper is to provide an overview of OHIO, present Eicon Technology's implementation of this interface and discuss various issues left unresolved by the current draft of OHIO as submitted to the IETF in April 1998.

Statement of Intent

As a recognized leader in providing Web-to-Host solutions, Eicon Technology has been actively involved in defining the emerging OHIO programming interface since its inception. Eicon Technology is committed to supporting OHIO and enhancing its implementation as the current definition evolves towards becoming an internationally accepted standard.

Evolution of OHIO

As was discussed earlier, OHIO was born of the consensus, from the industry's leading host-access vendors, that an open API was needed. A first draft of OHIO was proposed to the TN3270 working group of the IETF, and shortly thereafter an informal OHIO working

group was formed. In December 1998, the group proposed a definition for the API. The proposal was positively received, resulting in substantial progress being made toward defining a new open programming standard interface.

After the minutes of the first meeting were published, a revised draft of the OHIO proposal was issued. This incorporated feedback, detailed semantics and minor changes to the API. Major modifications that were agreed upon by the working group were not included. As such, the current official draft of OHIO is still far from a final definition of the API. Therefore, to date, no vendor can accurately claim to have a true working implementation of OHIO as it is still in its conception stage.

Overview of the OHIO API

The first draft of OHIO submitted to the IETF in December 1998 looks much like HLLAPI in an Object form; it is very 3270 centric.

> **"***OHIO addresses the need for a standardized advanced programming interface to access host data. OHIO does not modify the TN3270/TN5250 protocol or data stream but instead provides a common access method to that data once it arrives at the client. OHIO uses an Object Oriented approach to divide the data into logical objects, and provides methods on those objects to allow standard access to the data.* " *(<draft—ietf-tn3270e-ohio-01.txt.>, April 1999)*

Essentially, OHIO consists of defining the following objects and hierarchy:

```
OhioManager:  Contains one:
      OhioSessions:  Contains a collection of:
         OhioSession:  Contains one:
            OhioScreen:  Contains one of each of:
               OhioOIA:  The operator information area
               OhioFields:  Contains a collection of:
                  OhioField:  A field in the presentation space

Additional utility classes:
   OhioPosition
```

One of the goals of OHIO is to provide binary compatibility. This important characteristic will enable an application written and tested with one OHIO implementation to be used with any OHIO implementation, without recompilation.

Eicon Technology's Implementation of OHIO

In order to lead and speed up the definition of OHIO, Eicon has implemented the OHIO interface as defined by the current draft. For those customers who would like a heads-up on developing their own applications, Eicon is providing a first implementation of OHIO within its Aviva for Java product. Although modifications will be brought to the API as it evolves through a standard, applications can already be designed based on this implementation that has all the fundamental features of a host application interface.

Overview of Aviva for Java

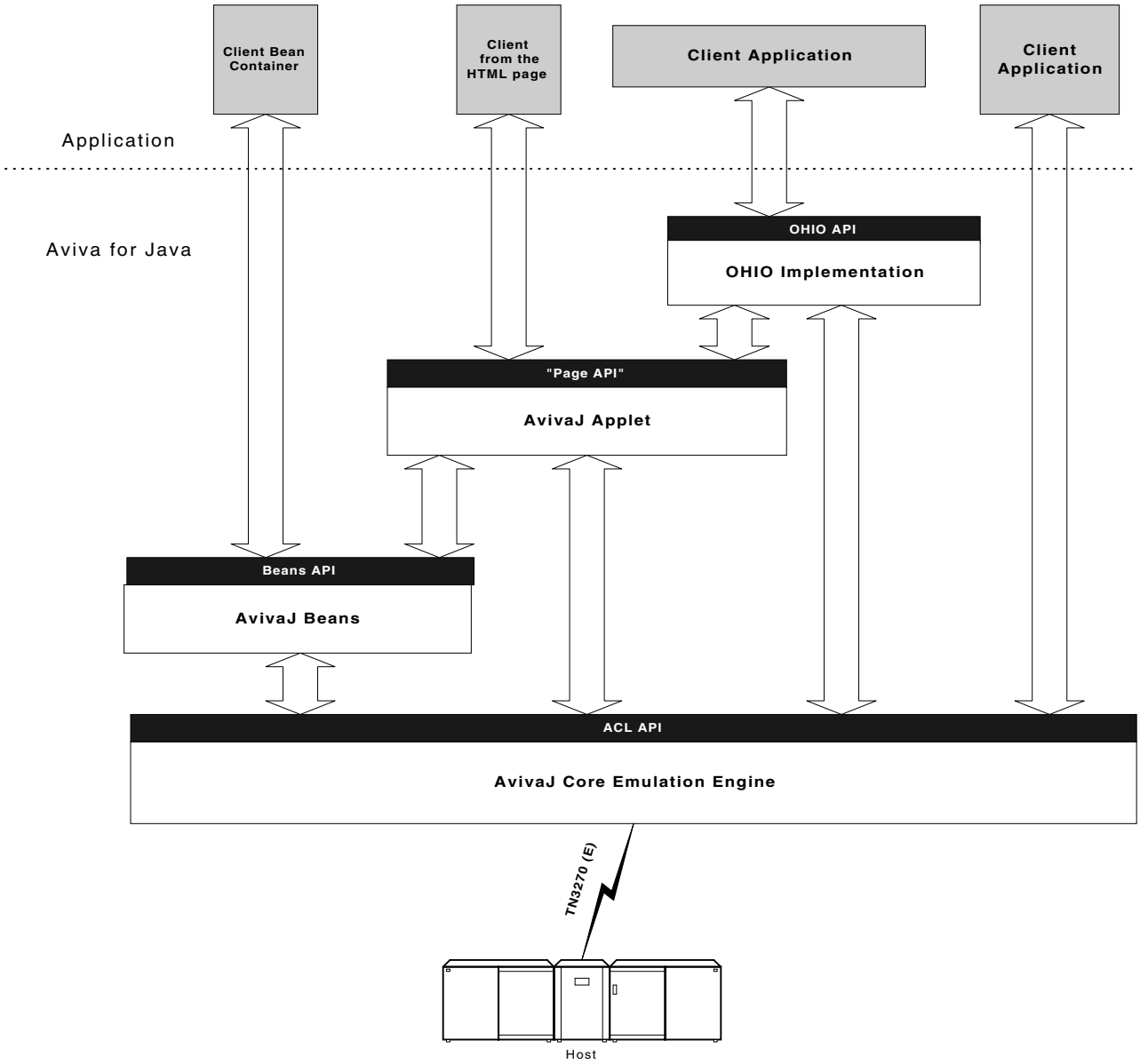
Eicon's Aviva for Java is a TN3270/TN3270E emulator Java applet. It is built around a core connectivity engine that exposes itself via a disciplined API known as **A**viva **C**lass **L**ibrary (ACL). The connectivity engine encapsulates the whole protocol stack up to and including 3270 Data Stream handling. ACL abstracts this functionality and allows programmers to build various ACL client applications. The Aviva for Java UI (User Interface) is one of many such applications.

Aviva for Java offers [MB1]other services [MB2]through another API known as "Page API". Applets and scripts residing in the same web page as Aviva for Java are given access to Aviva for Java specific services, including session management, user interface control, and full datastream access.

Aviva for Java also features a collection of Java Beans components. Just like ACL, they allow applications to be built around the Aviva for Java connectivity engine. Beyond basic connectivity, the beans provide a design-time environment that allow visual drag-and-drop programming using industry-standard tools such as Jbuilder, Visual Café or VisualAge. Furthermore, Java Beans also offer runtime User Interface capabilities, such as a functional emulator screen, which are easily incorporated into user applications. Using Java Beans significantly reduces the development cycle.

OHIO closely resembles Eicon's APIs. Actually, Eicon has built its implementation of OHIO on top of its Aviva Class Library.

The following diagram illustrates where OHIO resides within the current Aviva for Java API architecture. The OHIO implementation uses the services of ACL and of "Page API". ACL and Aviva for Java's Java Beans are "component APIs", i.e.: they allow new applications to incorporate those components. Page API is a "Control API" allowing other applications to control Aviva for Java. OHIO is a hybrid API permitting both types of services. To control Aviva for Java, OHIO uses Page API. To provide another API for the communications engine, OHIO uses ACL.



Objects Relationships

As per OHIO specifications (see ***Overview of OHIO API***), only the root object `com.eicon.iConnect.ohio.OhioManager` has to be instantiated. OHIO defines an ownership or containment hierarchy in which other objects cannot be instantiated, only retrieved from their containing object. The containment hierarchy is organized by a series of established rules.

- `OhioManager` has one or more OhioSessions collection objects obtained using ohioManager.getSessions().
- OhioSessions is a snapshot of existing OhioSession objects at the time ohioManager.getSessions() or ohioSessions.refresh() are called.

- OhioSessions is a collection of OhioSession objects obtained using ohioSessions.item() or ohioManager.openSession();

- OhioSession has one OhioScreen object obtained using ohioSession.getScreen();

- OhioScreen has one OhioOIA object obtained using ohioScreen.getOIA();

- OhioScreen has one or more OhioFields collection objects obtained using ohioScreen.getFields();

- OhioFields is a snapshot of existing OhioField objects at the time ohioScreen.getFields() or ohioFields.refresh() are called; OhioFields is a collection of OhioField objects obtained using ohioFields.item() or ohioFields.findByString(), ohioFields.findByPosition();

- OhioPosition can be obtained using createOhioPosition() on any object that implements iOhio: OhioSessions, OhioSession, OhioScreen, OhioOIA, OhioFields, OhioField.

The OHIO specification prescribes that objects that implement iOhio supply a getVendorObject() method, i.e.: OhioSessions, OhioSession, OhioScreen, OhioOIA, OhioFields, OhioField. It is up to the vendor to return null, or a vendor specific object that closely matches the functionality of the OHIO object involved.

Using getVendorObject() leads to incompatible code, therefore, it is only recommended when a vendor specific function is not supported by OHIO. As previously discussed in the "Overview of Aviva for Java" section, Eicon's implementation of OHIO relies upon ACL, making it a more complete offering. Eicon exposes ACL objects for which OHIO is a thin wrapper, in the following way:

- OhioSession returns ACLTerminal, OhioScreen returns ACLPS, OhioOIA returns ACLOIA, OhioFields returns ACLFields and OhioField returns ACLField.

The Relationship Between OHIO and Eicon's Aviva for Java Applet

The intent of this section is to clarify the two possible uses of the OHIO implementation within Eicon's Aviva for Java as: a) a component API, and b) an Aviva for Java control API.

This distinction is important and is influenced by the programmer's specific requirements.

1 – OHIO as a Component API

Consider being a Web application designer needing to connect to an IBM mainframe using a Java-based emulator. You are searching for a strong emulation engine with a library which takes care of all aspects of TN3270 connectivity. When selecting the appropriate emulator, you need to ask yourself the following questions:

a) *Is the emulator stand-alone or does it require the installation of some other product(s)?*
b) *If I so choose, can I remove the emulator's default UI and create my own? If errors occur, I want my application to be notified via return codes, exceptions, etc, and not have any message box invoked.*
c) *Is the emulator configuration agnostic or does it require a specific file in a specific directory to work? Only configuration agnostic components can seamlessly integrate with other configuration systems.*
d) *Is the API standard? That is, if this vendor's implementation is not responsive to my needs, can I easily switch to another vendor's solution?*

With Eicon's solution, the answer to all of these questions is 'Yes': Questions 'b', 'c' and 'd' because the OHIO specification had them in mind; question 'a' because of Eicon's careful design.

2 – OHIO as an Aviva for Java Control API

> *You are a satisfied Eicon Aviva for Java customer who enjoys its look and feel and configuration management. You have two new requirements: 1) to automate some repetitive tasks, like automatic login, and b) to further tailor Aviva for Java to your enterprise's needs without having to rewrite the whole emulator.*

OHIO can be used to control Aviva for Java sessions for each of these requirements. If used in a Virtual Machine (VM) in which Aviva for Java is not present (which means the ACL classes are imbedded in the application), only type 1 functions are available. If used in a web page as an applet along with Aviva for Java, both type 1 and 2 functions are simultaneously available.

What then is the benefit of ACL (type 1 operation) and Page API (type 2 operation)? ACL and Page API still offer more versatile and efficient services than OHIO as it is defined today. In particular, Page API permits UI modifications to Aviva for Java sessions, which OHIO does not.

Implementation Issues

OHIO is still a work-in-progress. Consequently, there are issues to be resolved and incorporated into the proposal before it can become an accepted RFC. In some cases, the semantics are not clearly defined, leaving room for interpretation. Where specifications are not clear, Eicon Technology has based its implementation on the result of discussions that occurred at the working group's last meeting in December 1998. Some such issues are discussed below.

Connect and Disconnect

OHIO defines the connect() and disconnect() methods along with two session states: CONNECTED and DISCONNECTED. It is known that for virtually any implementation, connect and disconnect may take a long time. If a connection error occurs during the process of connecting or while connected, OHIO offers no means to inform the user.

Eicon's and other vendor's implementations have the notion of at least 2 additional states: CONNECTING, DISCONNECTING . Eicon has documented how its OHIO implementation deals with these states.

Configuration

It is desirable for an OHIO client to dynamically construct a configuration resource, without going to any file, i.e.: when the application has its own storage schema that may be different from the OHIO implementation schema. In this case, the client application would extract the emulation information from its own storage, convert it to a format acceptable by the OHIO implementation and pass it to OHIO in openSession. Currently, the configurationResource parameter is a string. As configurationResource contents are up to each vendor, this parameter would be much better described by the more general Object. The implementation could query the object type and take different decisions, rather than parsing a string. A vendor may decide to accept a Properties Object as its configuration and the calling application could build it dynamically.

Send AID

OhioScreen provides the sendAid method, which allows sending integer values representing 3270 and 5250 "action" keys. Those integers should be part of a documented enumeration (in the iOhio interface, just like other such enumerations). This is missing from the current OHIO specification. It is also the main reason why binary compatibility cannot be achieved now and why the next version of OHIO, which will presumably fix this problem, will not be backward compatible. As Eicon's policy is not to interfere with the OHIO interfaces provided by the working group, for now the user will have to use the integer constants defined for ACL.

Blocking Calls

Blocking calls are useful because some systems (ASP, JavaScript) do not support callbacks. With very predictable host applications, many programmers prefer the linear logic of the cycle: type something, press enter, wait for screen, collect data. Eicon has made the recommendation to the OHIO working group that blocking versions of sendAid, sendKeys, connect and disconnect be added.

Fields Collection Semantics

The current OHIO specification states that the fields collection is a snapshot of the current screen fields structure at collection creation. The semantics concerning this topic are not clear and will need to be refined in order to use the *OhioField* and *OhioFields* objects efficiently and ensure compatibility with various implementations.

Error Handling

The OHIO specification does not address error handling or exceptions. Error handling is part of semantics and as such, implementers should not be left to improvise upon it. The next draft of OHIO should address this important part of the implementation.

One of the basic goals of OHIO is to be binary compatible. Eicon strongly endorses this objective. Given that the OHIO specification is still in draft form, Eicon has made certain assumptions, and has documented these assumptions accordingly within Aviva for Java itself.

Backward Compatibility

As indicated earlier, the current OHIO implementation is based on a preliminary and draft version of the OHIO specification. Once the final specification is released, OHIO will evolve with backward compatibility in mind. Backward compatibility with this preliminary version is not yet achievable, as some parameters may be modified and some others may be added or even removed.

As was previously mentioned, the current definition of OHIO does not yet include many of the changes that were agreed upon in the first committee meeting held in Orlando in December 1998. Therefore, method signatures, constant values and even class names may change.

To date, some semantics within the current OHIO draft remain undefined. These will surely be clarified in the final OHIO proposal and will be incorporated into Eicon's implementation of the OHIO API.

Conclusion

Eicon Technology, one of the pioneers in defining the OHIO standard, believes that OHIO will evolve into a flexible, contemporary API that is not restrictive to traditional HLLAPI features.

Eicon's first implementation of OHIO within Aviva for Java will help its customers to familiarize themselves with OHIO and allow them to start their Web-to-Host application development using a standard API. Furthermore, it will assist Eicon in identifying those areas where OHIO, or its implementation, needs improvement. In turn, Eicon will be able to provide this essential feedback to the official OHIO working group.

At such time when OHIO does become an approved standard, Eicon Technology will immediately update its implementation and promote OHIO as the standard API to be used for all applications accessing host information via the Internet.

References

1. Open Host Interface Objects for TN3270E, TN3270E Working Group Internet Draft: <draft-ietf-tn3270e-ohio-01.txt>, Expiration Date: October 1st, 1999, Thomas Brawn IBM Corporation, Stephen Gunn Attachmate Corporation. Distributed at ftp://service.boulder.ibm.com/software/standards/ohio/draft/draft-ietf-tn3270e-ohio-01.pdf.
2. IETF Ohio December 1998 meeting minutes distributed at ftp://service.boulder.ibm.com/software/standards/ohio/OrlandoMinutes.html.
3. Javadoc and Java sources distributed along with above document in April 1999 at ftp://service.boulder.ibm.com/software/standards/ohio/java/draft-ietf-tn3270e-ohio-01-java.zip.

###

Jacinthe Paradis is the Senior Program Manager for Eicon's Aviva Family of Next Generation Host Access products, which includes the most comprehensive range of PC-to-host access products for delivering host applications to the desktop or Web browser via SNA or IP centric networks.

Eugène Aresteanu is the Senior Software Architect of the Aviva products. He was a major contributor to the Aviva family architecture and the principal software architect of Aviva for Java. Eugène is a major contributor to the definition of OHIO and implemented it in Aviva for Java.