                     Open Host Interface Objects

Status of this Memo
   This document is an Internet-Draft and is in full conformance with
   all provisions of Section 10 of RFC2026.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as
   Internet-Drafts.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress." The list
   of current Internet-Drafts can be accessed at
      http://www.ietf.org/ietf/1id-abstracts.txt

   The list of Internet-Draft Shadow Directories can be accessed at
      http://www.ietf.org/shadow.html.

Abstract
   This draft addresses the need for a common, advanced, client
   programming interface to mainframe host data. Application
   developers, in particular third party application vendors, use
   API's provided by client emulator programs to write applications
   that access host data. Currently, the defacto standard HLLAPI is
   the interface most commonly provided by these programs and the one
   used by the third party application vendors.   However, HLLAPI is
   plagued by the fact that it doesn't exploit modern programming
   techniques and is fragmented by multiple, proprietary
   implementations. Client vendors have, over the last couple of years,
   developed more advanced interfaces that exploit modern programming
   advances. However no effort has been put into standardizing these
   interfaces and application developers have been forced to either
   chose to stay with HLLAPI or perform costly re-implementation of
   their products to support each competing advanced client API.

   The Open Host Interface Objects (OHIO) address the need for a
   standardized advanced programming interface to the host data. OHIO
   does not modify the TN3270/TN5250 protocol or datastream 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. Details about the connection protocol
used to communicate with the host and the specific host platform.

Table of Contents

1.0 Introduction

The following is the Ohio containment hierarchy:
   OhioManager: Contains one:
      OhioSessions: Contains a collection of1:
         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

The Ohio inheritance hierarchy is:
    Ohio: Base class
        OhioManager
        OhioSessions
        OhioSession
        OhioScreen
        OhioOIA
        OhioFields
        OhioField
        OhioPosition

2.0 Object Definitions

OMG IDL Version 2.1 is used to define the following objects. For a
mapping of the IDL to Java, see Appendix A - Ohio Java Mapping. For
a mapping of the IDL to ActiveX IDL, see Appendix B - Ohio ActiveX
IDL Mapping.

Note: "1" based counting is used throughout this document for both
positions (the first position on the screen is position 1, not
position 0) and sizes (the first item in a collection is item 1, not
item 0).

2.1 Ohio

Base class for all Ohio classes. Contains all common Ohio methods
and properties.

```
Interface Ohio {

    // This enum is used by:
    //    OhioFields.FindByString
    //    OhioScreen.FindString
    Readonly Attribute enum OHIO_DIRECTION {
        OHIO_DIRECTION_FORWARD  // Forward (beginning towards end)
        OHIO_DIRECTION_BACKWARD // Backward (end towards beginning)
    };

    // This enum is used by:
    //    OhioSession.sessionType
    Readonly Attribute enum OHIO_TYPE {
        OHIO_TYPE_UNKNOWN  // Unknown host
        OHIO_TYPE_3270     // 3270 host
        OHIO_TYPE_5250     // 5250 host
    };
```

```
          // This enum is used by:
          //    OHIOSession.connected
          //    OHIOSession sessionChanged
          Readonly Attribute enum OHIO_STATE {
             OHIO_STATE_DISCONNECTED  // The communication link to the
                                      // host is disconnected.
             OHIO_STATE_CONNECTED     // The communication link to the
                                      // host is connected.
          };

          // This enum is used by:
          //    OHIOField.getData
          //    OHIOScreen.getData
          Readonly Attribute enum OHIO_PLANE {
             OHIO_PLANE_TEXT   // Indicates Text Plane (character data)
             OHIO_PLANE_COLOR  // Indicates Color Plane (standard HLLAPI
                               // CGA color values)
             OHIO_PLANE_FIELD  // Indicates Field Attribute Plane (field
                               // attribute bytes)
             OHIO_PLANE_EXTENDED  // Indicates Extended Plane (extended
                                  // attribute bytes)
          };

          // These values are returned in the Ohio Color Plane from the
          // following methods:
          //    OhioField.getData
          //    OhioScreen.getData
          Readonly Attribute enum OHIO_COLOR {
             OHIO_COLOR_BLACK
             OHIO_COLOR_BLUE
             OHIO_COLOR_GREEN
             OHIO_COLOR_CYAN
             OHIO_COLOR_RED
             OHIO_COLOR_MAGENTA
             OHIO_COLOR_WHITE
             OHIO_COLOR_YELLOW
          };

          // These values are returned in the Ohio Extended Field Plane
          // from the following methods:
          //    OhioField.getData
          //    OhioScreen.getData
          Readonly Attribute enum OHIO_EXTENDED {
             OHIO_EXTENDED_HILITE   // Bitmask for Highlighting Bits
             OHIO_EXTENDED_COLOR    // Bitmask for Color Bits
             OHIO_EXTENDED_RESERVED // Bitmask for Reserved Bits
             OHIO_EXTENDED_HILITE_NORMAL  // Normal highlighting
             OHIO_EXTENDED_HILITE_BLINK  // Blinking highlighting
```

OHIO_EXTENDED_HILITE_REVERSEVIDEO // Reverse Video
                                   // highlighting

```
        OHIO_EXTENDED_HILITE_UNDERSCORE  // Underscore
                                         // highlighting
        OHIO_EXTENDED_COLOR_DEFAULT   // Default color
        OHIO_EXTENDED_COLOR_BLUE      // Blue
        OHIO_EXTENDED_COLOR_RED       // Red
        OHIO_EXTENDED_COLOR_PINK      // Pink
        OHIO_EXTENDED_COLOR_GREEN     // Green
        OHIO_EXTENDED_COLOR_TURQUOISE // Turquoise
        OHIO_EXTENDED_COLOR_YELLOW    // Yellow
        OHIO_EXTENDED_COLOR_WHITE     // White
    };

    // These values are returned in the Ohio Field Attributes from
    // the following methods:
    //    OhioScreen.getData
    Readonly Attribute enum OHIO_FIELD {
        OHIO_FIELD_ATTRIBUTE       // Bitmask for field attribute
        OHIO_FIELD_PROTECTED       // Protected field
        OHIO_FIELD_NUMERIC         // Numeric field
        OHIO_FIELD_PEN_SELECTABLE  // Pen selectable field
        OHIO_FIELD_HIGH_INTENSITY  // High intensity field
        OHIO_FIELD_HIDDEN          // Hidden field
        OHIO_FIELD_RESERVED        // Reserved field
        OHIO_FIELD_MODIFIED        // Modified field
    };

    // This enum is used by:
    //    OhioScreen event processing
    Readonly Attribute enum OHIO_UPDATE {
        OHIO_UPDATE_CLIENT  // Update initiated by client
        OHIO_UPDATE_HOST    // Update initiated by host
    };

    // This enum is used by:
    //    OhioOIA.owner
    Readonly Attribute enum OHIO_OWNER {
        OHIO_OWNER_UNKNOWN  // Uninitialized
        OHIO_OWNER_APP      // Application or 5250 host
        OHIO_OWNER_MYJOB    // 3270 - Myjob
        OHIO_OWNER_NVT      // 3270 in NVT mode
        OHIO_OWNER_UNOWNED  // 3270 - Unowned
        OHIO_OWNER_SSCP     // 3270 - SSCP
    };

    // This enum is used by:
    //    OhioOIA.InputInhibited
    Readonly Attribute enum OHIO_INPUTINHIBITED {
        OHIO_INPUTINHIBITED_NOTINHIBITED  // Input not inhibited
```

```
OHIO_INPUTINHIBITED_SYSTEM_WAIT   // Input inhibited by a
                         // System Wait state ("X SYSTEM" or "X []")
```

```
        OHIO_INPUTINHIBITED_COMMCHECK  // Input inhibited by a
                          // communications check state ("X COMMxxx")
        OHIO_INPUTINHIBITED_PROGCHECK  // Input inhibited by a
                          // program check state ("X PROGxxx")
        OHIO_INPUTINHIBITED_MACHINECHECK  // Input inhibited by a
                          // machine check state ("X MACHxxx")
        OHIO_INPUTINHIBITED_OTHER  // Input inhibited by something
                                    // other than above states
    };

    // The OHIO version level of this implementation.  The form is
    // "OHIO nn.nn"
    Readonly Attribute string OhioVersion;

    // The name of the vendor providing this OHIO implementation.
    // Format is vendor defined.
    Readonly Attribute string VendorName;

    // The vendor product version that is providing the OHIO
    // implementation.  Format is vendor specific.
    Readonly Attribute string VendorProductVersion;

    // The vendor object that provides non-standard, vendor
    // specific extensions to the OHIO object.
    Readonly Attribute Object VendorObject;

    // Create an OhioPosition object.
    //    row  The row coordinate.
    //    col  The column coordinate.
    OhioPosition CreateOhioPosition(row, col);

};
```

2.2 OhioField

A field is the fundamental element of a virtual screen. A field
includes both data and attributes describing the field. The
OhioField class encapsulates a virtual screen field and provides
methods for accessing and manipulating field attributes and data.

OhioField objects can be accessed only through the OhioFields
object.

```
Interface OhioField {

    // The starting position of the field.  The position can range
    // from 1 to the size of the virtual screen.  The starting
    // position of a field is the position of the first character
```

```
// in the field.
Readonly Attribute OhioPosition Start;
```

```
            // The ending position of the field.  The position can range
            // from 1 to the size of the virtual screen.  The ending
            // position of a field is the position of the last character in
            // the field.
            Readonly Attribute OhioPosition End;

            // The length of the field.  A field's length can range
             from 1 to the size of the
            // virtual screen.
            Readonly Attribute int Length;

            // The attribute byte for the field.
            Readonly Attribute int Attribute;

            // Indicates whether or not the field has been modified.  True
            // if the field has been modified, otherwise false.
            Readonly Attribute boolean Modified;

            // Indicates whether or not the field is protected.  True if
            // the field is protected, otherwise false.
            Readonly Attribute boolean Protected;

            // Indicates whether or not the field is numeric-only.  True if
            // the field is numeric only, otherwise false.
             Readonly Attribute boolean Numeric;

            // Indicates whether or not the field is high-intensity.  True
            // if the field is high intensity, otherwise false.
            Readonly Attribute boolean HighIntensity;

            // Indicates whether or not the field is pen-selectable.  True
            // if the field is pen-selectable, otherwise false.
            Readonly Attribute boolean PenSelectable;

            // Indicates whether or not the field is hidden.  True if the
            // field is hidden, otherwise false.
            Readonly Attribute boolean Hidden;

            // The text plane data for the field.  This is similar to the
            // getData() method using the OHIO_PLANE_TEXT parameter, except
            // the data is returned as a string instead of a character
            // array.  When setting the String property, if the string is
            // shorter than the length of the field, the rest of the field
            // is cleared.  If the string is longer than the field, the
            // text is truncated.  A subsequent call to this property will
            // not reflect the changed text.  To see the changed text, do a
            // refresh on the OhioFields collection and retrieve a new
            // OhioField object.
            Attribute string String;
```

```
        // Returns data from the different planes (text, color,
        // extended) associated with the field.  The data is returned
        // as a character array.
        //    targetPlane  An OHIO_PLANE value indicating from which
        //                 plane to retrieve the data.
        char[] getData(OHIO_PLANE targetPlane);

    };
```

   2.3 OhioFields

      OhioFields contains a collection of the fields in the virtual
      screen. It provides methods to iterate through the fields, find
      fields based on location, and find fields containing a given
      string. Each element of the collection is an instance of
      OhioField.

      OhioFields can only be accessed through OhioScreen using the
      Fields property. OhioFields is a static view of the virtual
      screen and does not reflect changes made to the virtual screen
      after its construction. The field list can be updated with a new
      view of the virtual screen using the Refresh() method.

      Note: All OhioField objects returned by methods in this class are
      invalidated when Refresh() is called.

```
      Interface OhioFields {

         // Returns the number of OhioField objects contained in this
         // collection.
         Readonly Attribute int Count;

         // Returns the OhioField object at the given index.  "One
         // based" indexing is used in all Ohio collections.  For
         // example, the first OhioField in this collection is at
         // index 1.
         OhioField Item(int fieldIndex);

         // Updates the collection of OhioField objects.  All OhioField
         // objects in the current virtual screen are added to the
         // collection.  Indexing of OhioField objects will not be
         // preserved across refreshes.
         void Refresh();

         // Searches the collection for the target string and returns
         // the OhioField object containing that string.  The string
         // must be totally contained within the field to be considered
         // a match.  If the target string is not found, a null will be
```

```
        // returned.
        //    targetString  The target string.
```

```
        //    startPos      The row and column where to start.  The
        //                  position is inclusive (for example, row 1,
        //                  col 1 means that position 1,1 will be used
        //                  as the starting location and 1,1 will be
        //                  included in the search).
        //    length        The length from startPos to include in the
        //                  search.
        //    dir           An OHIO_DIRECTION value.
        //    ignoreCase    Indicates whether the search is case
        //                  sensitive.  True means that case will be
        //                  ignored.  False means the search will be
        //                  case sensitive.
        OhioField FindByString(String targetString,
                               OhioPosition startPos,
                               int length,
                               OHIO_DIRECTION dir,
                               boolean ignoreCase);

        // Searches the collection for the target position and returns
        // the OhioField object containing that position.  If not
        // found, returns a null.
        //    targetPosition  The target row and column.
        OhioField FindByPosition(OhioPosition targetPosition);

     };


  2.4  OhioManager

     The central repository for access to all OHIO sessions.  The
     OhioManager contains a list of all OhioSession objects available
     on this system.

     Interface OhioManager {

        // An OhioSessions object containing the OhioSession objects
        // available on this system.  This list of objects is a static
        // snapshot at the time the OhioSessions object is created.
        // Use the OhioSessions.refresh method to obtain a new
        // snapshot.
        Readonly Attribute OhioSessions Sessions;

        // Returns an OhioSession object based on the search parameters
        // provided.
        //    ConfigurationResource  A vendor specific string used to
        //                           provide configuration information.
        //    SessionName  The unique name associated with an
        //                 OhioSession.
```

```
        // The parameters are used as follows:
        //    Is ConfigurationResource provided?
        //        Yes - Is SessionName provided?
        //            Yes - Is OhioSession object with matching
        //                    SessionName available on the system?
        //                Yes - Error, attempting to create an
        //                        OhioSession object with a non-unique
        //                        SessionName.
        //                No - Create an OhioSession object using
        //                        SessionName and ConfigurationResource.
        //            No - Start a new OhioSession using
        //                    ConfigurationResource and generating a new
        //                    SessionName.
        //        No - Is SessionName provided?
        //            Yes - Is OhioSession object with matching
        //                    SessionName available on the system?
        //                Yes - Return identified OhioSession object.
        //                No - Return null.
        //            No - Return null.
        OhioSession OpenSession(String ConfigurationResource,
                                String SessionName);

        // Closes an OhioSession object.  The OhioSession is
        // considered invalid and is removed from the list of
        // OhioSession objects.
        //    SessionObject The OhioSession to close.
        void CloseSession(OhioSession SessionObject);

        // Closes an OhioSession object.  The OhioSession is
        // considered invalid and is removed from the list of
        // OhioSession objects.
        //    SessionName The SessionName of the OhioSession to
        //                close.
        void CloseSession(String SessionName);

    };
```

2.5 OhioOIA

The operator information area of a host session. This area is
used to provide status information regarding the state of the host
session and location of the cursor.

An OhioOIA object can be obtained using the GetOIA() method on an
instance of OhioScreen.

```
    Interface OhioOIA {
```

```
     // Indicates whether the field which contains the cursor is an
     // alphanumeric field.  True if the cursor is in an
     // alphanumeric field, false otherwise.
     Readonly Attribute boolean Alphanumeric;

     // The communication check code.  If InputInhibited returns
     // OHIO_INPUTINHIBITED_COMMCHECK, this property will return the
     // communication check code.
     Readonly Attribute int CommCheckCode;

     // Indicates whether or not input is inhibited.  If input is
     // inhibited, SendKeys or SendAID calls to the OhioScreen are
     // not allowed.  Why input is inhibited can be determined from
     // the value returned.  If input is inhibited for more than one
     // reason, the highest value is returned.
     Readonly Attribute OHIO_INPUTINHIBITED InputInhibited;

     // The machine check code.  If InputInhibited returns
     // OHIO_INPUTINHIBITED_MACHINECHECK, this property will return
     // the machine check code.
     Readonly Attribute int MachineCheckCode;

     // Indicates whether the field which contains the cursor is a
     // numeric-only field.  True if the cursor is in a numeric-only
     // field, false otherwise.
     Readonly Attribute boolean Numeric;

     // Indicates the owner of the host connection.
     Readonly Attribute OHIO_OWNER Owner;

     // The program check code.  If InputInhibited returns
     // OHIO_INPUTINHIBITED_PROGCHECK, this property will return the
     // program check code.
     Readonly Attribute int ProgCheckCode;

   };


   2.6  OhioPosition

   Holds row and column coordinates.  An OhioPosition can be
   constructed by using CreateOhioPosition() on any Ohio class.

   Interface OhioPosition {

     // The row coordinate
     Attribute int Row;

     // The column coordinate.
```

```
        Attribute int Column;

    };
```

2.7 OhioScreen

OhioScreen encapsulates the host presentation space. The
presentation space is a virtual screen which contains all the
characters and attributes that would be seen on a traditional
emulator screen. This virtual screen is the primary object for
text-based interactions with the host. The OhioScreen provides
methods that manipulate text, search the screen, send keystrokes
to the host, and work with the cursor.

An OhioScreen object can be obtained from the Screen property of
an instance of OhioSession.

The raw presentation space data is maintained in a series of
planes which can be accessed by various methods within this class.
The text plane contains the actual characters in the presentation
space. Most of the methods in OhioScreen class work exclusively
with the text plane.

The remaining planes contain the corresponding attributes for each
character in the text plane. The color plane contains color
characteristics. The field plane contains the field attributes.
The extended plane contains the extended field attributes. The
color, field, and extended planes are not interpreted by any of
the methods in this class.

Interface OhioScreen {

   // The location of the cursor in the presentation space. The
   // row and column of the cursor is contained within the
   // OhioPosition object.
   Attribute OhioPosition Cursor;

   // The OhioOIA object associated with this presentation space.
   // This object can be used to query the status of the operator
   // information area.
   Readonly Attribute OhioOIA OIA;

   // The OhioFields object associated with this presentation
   // space. This provides another way to access the data in the
   // virtual screen. The OhioFields object contains a snapshot
   // of all the fields in the current virtual screen. Fields
   // provide methods for interpreting the data in the non-text
   // planes. Zero length fields (due to adjacent field
   // attributes) are not returned in the OhioFields collection.
   // For unformatted screens, the returned collection contains
   // only one OhioField that contains the whole virtual screen.
   Readonly Attribute OhioFields Fields;

```
   // The number of rows in the presentation space.
   Readonly Attribute int Rows;
```

```
// The number of columns in the presentation space.
Readonly Attribute int Columns;

// The entire text plane of the virtual screen as a string.
// All null characters and Field Attribute characters are
// returned as blank space characters.
Readonly Attribute string String;

// Returns a character array containing the data from the Text,
// Color, Field or Extended plane of the virtual screen.
//    start  The row and column where to start.  The position
//           is inclusive (for example, row 1, col 1 means that
//           position 1,1 will be used as the starting location
//           and 1,1 will be included in the data).  "start"
//           must be positionally less than "end".
//    end    The row and column where to end.  The position is
//           inclusive (for example, row 1, col 1 means that
//           position 1,1 will be used as the ending location
//           and 1,1 will be included in the data). "end" must
//           be positionally greater than "start".
//    plane  A valid OHIO_PLANE value.
char[] getData(OhioPosition start,
               OhioPosition end,
               OHIO_PLANE plane);

// Searches the text plane for the target string. If found,
// returns an OhioPosition object containing the target
// location.  If not found, returns a null.  The targetString
// must be completely contained by the target area for the
// search to be successful.  Null characters in the text plane
// are treated as blank spaces during search processing.
//    targetString  The target string.
//    startPos  The row and column where to start.  The
//              position is inclusive (for example, row 1,
//              col 1 means that position 1,1 will be used as
//              the starting location and 1,1 will be included
//              in the search).
//    length  The length from startPos to include in the
//            search.
//    dir  An OHIO_DIRECTION value.
//    ignoreCase  Indicates whether the search is case
//                sensitive.  True means that case will be
//                ignored.  False means the search will be case
//                sensitive.
OhioPosition FindString(string targetString,
                        OhioPosition start,
                        int length,
                        int OHIO_DIRECTION,
```

```
                        boolean ignoreCase);
```

```
    // The sendKeys method sends a string of keys to the virtual
    // screen.  This method acts as if keystrokes were being typed
    // from the keyboard.
    //
    // The keystrokes will be sent to the location given.  If no
    // location is provided, the keystrokes will be sent to the
    // current cursor location.
    //    text  The string of characters to be sent.
    void sendKeys(string text,
                    OhioPosition location);

    // The sendAid method sends an "aid" keystroke to the virtual
    // screen.  These aid keys can be though of as special
    // keystrokes, like the Enter key, the Tab key, or the Page Up

    // key.  All the valid special key values are contained in the
    // OHIO_AID enumeration.
    //    aidKey  The aid key to send to the virtual screen.
    void sendAid(OHIO_AID aidKey);

    // The putString method sends a string to the virtual screen at
    // the specified location.  The string will overlay only
    // unprotected fields, and any parts of the string which fall

    // over protected fields will be discarded.
    //    text  String to place in the virtual screen.
    //    location  Position where the string should be written.
    void setString(string text,
                    OhioPosition location);

};


2.8  OhioSession

A host session.

Interface OhioSession {

    // The configurationResource for this OhioSession object.
    Readonly Attribute string ConfigurationResource;

    // Indicates whether this OhioSession object is connected to a
    // host.  True means connected, false means not connected.
    Readonly Attribute boolean Connected;

    // The SessionName for this OhioSession object.  The
    // SessionName is unique among all instances of OhioSession.
    Readonly Attribute string SessionName;
```

```
        // The SessionType for this OhioSession object.
        Readonly Attribute OHIO_TYPE SessionType;
```

```
        // The OhioScreen object for this session.
        Readonly Attribute OhioScreen Screen;

        // Starts the communications link to the host.
        void Connect();

        // Stops the communications link to the host.
        void Disconnect();

    };
```

2.9 OhioSessions

Contains a collection of OhioSession objects. This list is a
static snapshot of the list of OhioSession objects available at
the time of the snapshot.

```
    Interface OhioSessions {

        // The number of OhioSession objects contained in this
        // collection.
        Readonly Attribute int Count;

        // The OhioSession object at the given index.  "One based"
        // indexing is used in all Ohio collections.  For example, the
        // first OhioSession in this collection is at index 1.
        //    index  The index of the target OhioSession.
        OhioSession Item(int index);

        // The OhioSession object with the given SessionName.  Returns
        // null if no object with that name exists in this collection.
        //    SessionName  The target name.
        OhioSession Item(string SessionName);

        // Updates the collection of OhioSession objects.  All
        // OhioSession objects that are available on the system at the
        // time of the refresh will be added to the collection.
        // Indexing of OhioSession objects will not be preserved across
        // refreshes.
        void Refresh();

    };
```

3.0 Acknowledgments

This document was initiated in the TN3270E Working Group and was
completed by a dedicated group of interested parties.

The authors wish to thank the following individuals for their
contributions to this document:
- Jeffrey Altman, Columbia University
- Eugene Aresteanu, Eicon Technology
- Blair Cooper, Attachmate Corporation
- Andrew Duckworth, Platypus Partners, Inc.
- Brian L. Henry, Attachmate Corporation
- Gary Horen, WRQ
- Pierre Goyette, Hummingbird Communications, Ltd.
- Greg Knowles, IBM Corporation
- Warren Mackensen, NetManage, Inc.
- Mark McMillan, IBM Corporation
- Hemant Nanivadekar, Attachmate Corporation
- J. Burke Ryder, Attachmate Corporation
- Brian Webb, IBM Corporation

4.0 References

  - International Business Machines Corporation, "eNetwork Personal
    Communications Version 4.2 for Windows 95 and Windows NT Host
    Access Class Library", September 1997.
  - International Business Machines Corporation, "Host On-Demand
    Version 2.0 Host Access Class Library for Java Reference",
    October 1997.
  - Attachmate Corporation, "EXTRA! Objects SDK Reference Guide for
    C++", October 1997.

5.0 How to Contact the Authors

    Thomas Brawn
    IBM Corporation
    4205 S. Miami Blvd
    RTP, NC 27709
    e-mail: brawntj@us.ibm.com
    phone: 919-254-8301

    Stephen Gunn
    Attachmate Corporation
    3617 131st Ave. S.E.
    Bellevue, Washington 98006
    e-mail: stevegu@attachmate.com
    phone: 425-649-6221

Appendix A - Ohio Java Mapping

Available using anonymous ftp from ftp.boulder.ibm.com. Once
logged in, change to the software/standards/ohio/java directory
to download the Java classes.

Appendix B - Ohio ActiveX IDL Mapping

   Available using anonymous ftp from ftp.boulder.ibm.com. Once
   logged in, change to the software/standards/ohio/idl directory to
   download the idl file.

Appendix C - 3270 Format Control Orders

   This appendix specifies the representation of 3270 Format Control
   Orders within OHIO text strings.

   The following special control codes are classified as 3270 format
   control orders. (This information is excerpted from the IBM 3270
   Information Display System Data Stream Programmer's Reference,
   Document Number GA23-0059-07.) On a 3270 display device, these
   format control orders are stored in the device buffer and
   displayed as shown below. Their representation in OHIO text
   strings (OhioScreen and OhioField) is included in the ANSI and
   UNICODE columns. The ANSI values were taken from the IBM
   Character Data Representation Library Character Data
   Representation Architecture Reference and Registry, Document
   Number SC09-2190-00. The UNICODE values were taken from
   ADDITIONAL CONTROL PICTURES FOR UNICODE,
   ftp://kermit.columbia.edu/kermit/ucsterminal/control.txt. Some
   of these values are proposed extensions to the Unicode Control
   Pictures.

   Abbr Order   EBCDIC ANSI Unicode Displayed As
.
   NUL Null   00   00   2400   A blank, suppressed on
                                                Read Modified
   SUB Substitute   3F   1A   241A   A solid circle
   DUP Duplicate   1C   1C   E07B   An overscore asterisk
   FM   Field Mark   1E   1E   E07D   An overscore semicolon
   FF   Form Feed   0C   0C   240C   A blank
   CR   Carriage Return 0D   0D   240D   A blank
   NL   New Line   15   85   2424   A blank
   EM   End of Medium   19   19   2419   A blank
   EO   Eight Ones   FF   9F   E07F   A blank

   Notes:
      ANSI 85: Conflicts with "Next Line" C1 control character (ISO
               6429).
      ANSI 9F: Conflicts with "Application Program Command" C1 control
               character (ISO 6429).

   NUL is read back (by the host) as a null (x'00') on a Read Buffer
   operation, but not read back on Read Modified operations.

NL, EM, FF, and CR are printer control codes with no display
function. However, the code must be supported to the extent of
being accepted and, on reading back, must appear as NL, EM, FF,
and CR respectively. All are displayed as a space.

FM and DUP are displayed as above. When read back, they appear
as the FM and DUP codes.
FM and DUP can be entered from the keyboard. They are stored in
the display buffer as controls; the current character set
selection has no effect on them. They are transmitted to the
application program as control codes.

The SUB local function, Error Override, entered from the
keyboard, is required only as a part of Field Validation.