



An SBA 8(a) Certified Business

Electronic Handbooks (EHBs)

Phase II Interim Report

PREPARED BY:

CONEY ISLAND INCORPORATED

2939 WEST 29 TH STREET

BROOKLYN, NY 11236-3030

PHONE: (718) 999-9999

FAX: (718) 999-8888

Email: nfrankfurter@coneyisland.com

Home Page: <http://coneyisland.com>

Contract Number: NAS13-04954

Report Number: CII-CR-2002-2234

January 2002

THESE SBIR DATA ARE FURNISHED WITH SBIR RIGHTS UNDER CONTRACT NO. NAS13-04954 FOR A PERIOD OF 4 YEARS AFTER ACCEPTANCE OF ALL ITEMS TO BE DELIVERED UNDER THIS CONTRACT. THE GOVERNMENT AGREES TO USE THESE DATA FOR GOVERNMENT PURPOSE ONLY, AND THEY SHALL NOT BE DISCLOSED OUTSIDE THE GOVERNMENT (INCLUDING DISCLOSURE FOR PROCUREMENT PURPOSE) DURING SUCH PERIOD WITHOUT PERMISSION OF THE CONTRACTOR, EXCEPT THAT, SUBJECT TO THE FOREGOING USE AND DISCLOSURE PROHIBITIONS, SUCH DATA MAY BE DISCLOSED FOR USE BY SUPPORT CONTRACTORS. AFTER THE AFORESAID 4-YEAR PERIOD THE GOVERNMENT HAS A ROYALTY-FREE LICENSE TO USE, AND TO AUTHORIZE OTHERS TO USE ON ITS BEHALF, THESE DATA FOR GOVERNMENT PURPOSE, BUT IS RELIEVED OF ALL DISCLOSURE PROHIBITIONS AND ASSUMES NO LIABILITY FOR UNAUTHORIZED USE OF THESE DATA BY THIRD PARTIES. THIS NOTICE SHALL BE AFFIXED TO ANY REPRODUCTIONS OF THESE DATA, IN WHOLE OR IN PART.

Project Summary

Electronic Handbooks (EHBs) are Internet-based tools that support the paperless documentation and management of complex distributed processes (e.g., Grants/Contracts Management). Tools include user interface and/or backend software. Processes are represented as "Internet-based plays" where "actors" communicate thru the Internet. For each role, EHBs guide actors thru their parts. Two applications of EHBs are: National Aeronautics and Space Administration's (NASA) Small Business Innovation Research (SBIR) Program, and Department of Justice's Bulletproof Vests Partnership (BVP) Program.* EHBs, like plays, are developed in six stages: Outline or Playwriting, Example or Rehearsal, Implementation or Staging, Utilization or Performance, Revision or New Production, and Cross-Subprocess or Cross-Play Analysis.

The goal of the project is to develop and implement an OLE for Process Control (OPC) based Quality of Service network protocol that will support fast communication and data transfers. The OPC network is to be used by network intelligent devices such as sensors and controllers. The network will be operable independent of network hardware and provide a flexible plug and play solution while reducing development and maintenance costs for hardware and software.

During this reporting period COM/OLE based in-proc servers, local servers and remote servers were studied. These technologies represent fundamental components of the OPC server design. The server design was performed using Object-Oriented Design (OOD). Quality of Service implementation strategies to be used in distinguishing traffic with immediate process requirements needs versus traffic able to tolerate latency have been discussed. Methods of managing bandwidth as well as traffic congestion control measures and general data handling procedures have been studied. The final goal is the implementation of QoS measures able to generate predictable data transfers while measuring latency and eliminating data loss.

During the next reporting period the development of the OPC server will continue and the OPC client development will be initialized. Work related to QoS analysis and bandwidth adaptation including protocol rules and definitions will begin.

Table of Contents

	<u>PAGE</u>
Cover	
Project Summary	
Table of Contents	
Table of Figures	
1 Identification and Significance of the Innovation	
2 Technical Objectives, Approach, and Work Plan	
2.1 Technical Objectives	
2.2 Approach	
2.3 Work Plan	
3 NASA and Non-NASA Applications	
3.1 NASA Applications	
3.2 Non-NASA Commercial Applications	
4 Development Team*	
4.1 Key Contractor Participants	
4.2 Key NASA Participants	
4.3 NASA and Non-NASA Advisors	
5 Technical Activities	
5.1 Recent Technical Activities	
5.2 Future Technical Activities	
6 NASA and Non-NASA Commercialization Activities	
6.1 Recent NASA Potential Customer Activities	
6.2 Recent Non-NASA Potential Customer Activities	
6.3 Other Recent Commercialization Activities	
6.4 Future Commercialization Activities	
7 Resources Status	
8 References	16

Table of Figures

Figure 2-1	The Beach	2
Figure 2-2	Luna Park.....	3
Figure 2-3	Wonder Wheen	4

1 Identification and Significance of the Innovation

This report is the first bimonthly report for the NASA SBIR Phase I Project: “Network Implementations for Real Time Communication and Data Transfer” (Contract No. NAS13-02001), undertaken by the American GNC (AGNC) Corporation, Simi Valley, CA. The contract is scheduled to be completed May 3, 2002.

The goal of the project is to develop and implement the OPC based Quality of Service network protocols to support the fast communication and data transfer for use by networked intelligent devices such as sensors and controllers. The OPC network is to be operable independent of network hardware and provide a flexible plug and play solution while reducing development and maintenance costs for hardware and software.

The objectives of *Electronic Handbooks* (EHBs) are to **Facilitate paperless documentation and management of complex distributed processes. Facilitate system development: requirements capture, system design, implementation,**

- **multi-developer coordination,**
 - **software distribution,**
 - **end-user learning,**
 - **system documentation,**
 - **system revisions, and**
 - **system reuse for similar processes.**
- **Facilitate process and system improvement.**

As an entry point of our OPC network strategy development, the OPC Specification is briefly discussed as part of this report. Industry QoS implementations and standards are briefly outlined and the QoS strategies to be used in the OPC network are discussed.

2 Technical Objectives, Approach, and Work Plan

2.1 Technical Objectives

The objectives of *Electronic Handbooks* (EHBs) are to **Facilitate paperless documentation and management of complex distributed processes. Facilitate system development: requirements capture, system design, implementation,**

- **multi-developer coordination,**
 - **software distribution,**
 - **end-user learning,**
 - **system documentation,**
 - **system revisions, and**
 - **system reuse for similar processes.**
- **Facilitate process and system improvement.**

Within the typical OPC network implementation, a client application can control and manage communication among different terminal devices (OPC servers) from a unique or multiple source manufacturers, to perform specific functions. OPC networks connect different terminal devices such as PLCs, sensors and controllers into a network.

Figure 2.1-1 illustrates the standard OPC Network Architecture.

Figure 2.1-1 The Beach

The implementation of the OPC standard uses the client/server architecture using OLE/COM, as shown in



Figure 2.1-2. The OPC Server samples data at high data rates from data sources such as sensors and other such instruments. The data collected can be real data such as temperature, pressure, speed, or status information regarding the physical local hardware connection as well as sub-system information. The OPC Client is manned from a control center and used to access and manage the data.

The OPC Client is the nerve center of this network protocol. The characteristic of our design and implementation is that the communication traffic monitoring and control, QoS management and supply, and bandwidth adaptation are all achieved on the client side in our architecture.

2.2 Approach

At a high level, an OPC Data Access Server is organized as several objects: the server, the groups, and the items. The OPC server object maintains the information about itself and serves as a container for OPC group objects. The OPC group object maintains information about itself and provides the mechanism for containing and logically organizing OPC items. The OPC Groups provide a way for clients to organize data. An item is the basic operational unit in OPC. The items in the same Group share the same properties, such as communication priority, data update rate, and transportation latency. It is worth of mentioning that our primary idea is that we can tune the properties of the OPC Group to implement the required QoS and bandwidth

adaptation. QoS and bandwidth are imperative aspects to be considered for the performance of the OPC based network protocol such as data rate, throughput, transport latency, etc. Therefore QoS will receive special consideration during the OPC server and client design.

Based on the discussion above, in this first period, we have concentrated on the OLE/COM component design and the principles of QoS used for OPC based network protocol implementation. Our overall design picture is shown in Figure 0-1. Figure 0-1 illustrates all key OPC technologies in such as local and remote server design, custom interface, automation wrapper and automation interface and shows that all technologies originated from the Microsoft's OLE/COM. This technology represents the building ground of the OPC technology.

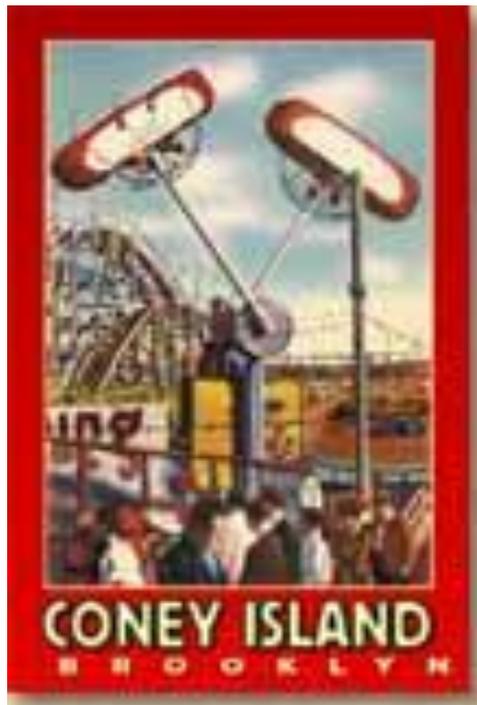


Figure 2.1-2 Luna Park

2.3 Work Plan

Some of the basic COM components will be designed and implemented using Object – Oriented Design (OOD) approach in this chapter which could be expanded for use in the further OPC design.

Object Linking and Embedding (OLE) is a well-defined set of COM-based interfaces and a set of API functions to facilitate the use of these interfaces. It provides a robust and application-level implementation of the COM model that can benefit the developer to build the reusable software. In other words, OLE changes the function-oriented APIs into object-oriented, robust and portable Object Oriented (OO) interfaces. An interface is a set of the methods that operates on the object that exposes them. There are two kinds of interfaces, standard and custom (provided by the developers or some organizations). The user can access the object's data only through its interface methods that are similar to accessing the private data member in C++.

In COM, the interface methods are accessible only via a pointer to the interface on the object. The interface methods are organized into the Vee-Table (VTBL). The rule there is that when a user has a pointer to an interface on an object, he can call that interface's methods, but he may not know the organization inside the object. The relationship between OLE and COM is shown in **Error! Reference source not found.**

The classes CClassFactory and CTestData inherit the public class interfaces IClassFactory and IDataObject, respectively. Class CClassFactory object needs to use CTestData class so that it can create a pointer to instance of CTestData and return it to the client for access. Class CTestData delegates the public class IDataAdvanceHolder which support the asynchronous data transfer. Once a client gets a pointer to a data object, it can subscribe to be notified when the server object changes the data. It is called exception-based data transfer in OPC server. The difference between in-proc server design and local server design is that for in-proc server, the file including DLLMain() function needs to provide several APIs for OLE to call as well as registering the class ID (CLISD) with the system; for local server, the file including WinMain() needs to initialize the OLE library and registers with the system once launched as a OLE server. The in-proc server and local server accesses are shown in Figure 3-3 and Figure 3-4.

From Figures 3-3 and 3-4, we can see that the connection between OLE and in-proc server is different from that between OLE and local server. Once a client calls CoCreateInstance() API, OLE reads registry and tries to find the In-proc server first, then a local server if the former is

not available. DLL is loaded if in-proc server is found. OLE calls API function DllGetClassObject() to obtain the pointer to the class factory in DLL, then calls IclassFactory::CreateInstance() to return client a pointer to data object. In the local server, OLE calls WinExec() and local server is launched if “-Embedding “ is on the command line. The server creates a class factory object and calls API function CoRegisterClassObject() to register the class factory. In this way, OLE can call the interface method IClassFactory::CreateInstance().

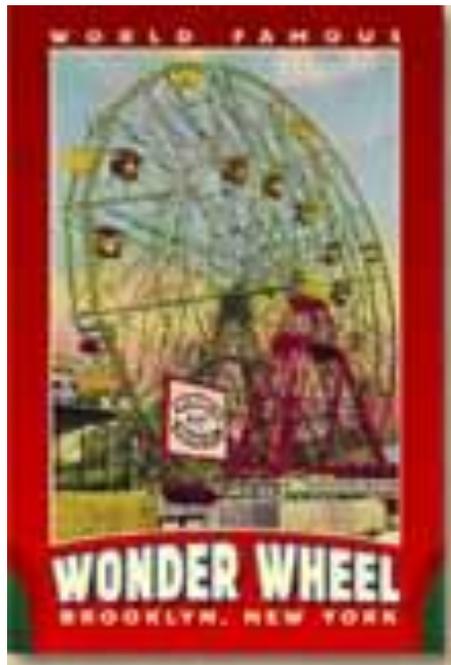


Figure 0-1 Wonder Wheel

OLE automation is a process where the automation server or DLL can create and expose automation objects to be used by the automation controller. Not only can automation objects expose the methods that can accept the arguments and return the values, but it can also expose the properties that are the data values of the objects, which can be set and gotten by the controller. The controller can access the methods and properties of the automation object only through the single OLE interface IDispatch with the following method:

```
IDispatch::GetIDsOfName // Map a single name into the DISPID
```

```
IDispatch::GetTypeInfo           // Get TypeID interface
IDispatch::GetTypeInfoCount      // Get number of the TypeID interfaces
IDispatch::Invoke                // used to access the methods and properties of the automation object
```

Actually only method `IDispatch::invoke` is used by the controller to access the methods and data of the object. Each method or property in the automation object is identified by the single unique ID call dispatch ID, which is a input parameter the controller needs to call the `IDispatch::invoke()` method.

Server design is similar to that in the last section in **Error! Reference source not found.** The difference is that instead of inheriting the `IDataObject` interface, the `CTestData` Class derives the `IDispatch` interface to provide a way for the controller to access the methods and properties of the object. The server could be in-proc and local.

In the controller side, a wrapper is built for application to access the `IDispatch` interface. The design is shown in Figure 3-5. The `CWrapper` class has a point to `IDispatch` interface as a member. It encapsulates the functions provided by the non-Object Oriented (OO) APIs such as `CoCreateInstance()` and the methods of the interface `IDispatch` within concise, robust and portable class interface. So controller can conveniently use this class to get or set the data in the automation object and access its methods.

Distributed COM (DCOM) is a software component provided by the operating system which provides the functionality that a client neither knows nor cares whether the server is in-proc, local or remote and a object server does not know if its clients are local or remote.

Remote server is similar to local server in design. For a remote server access, a client does something a little different. When creating instance of object on remote server, it needs to pass the “serverinfo” structure, remote server flag etc. to the API function `CoCreateInstanceEx()`. Also a client needs to have permission from the security system of the remote server. For a remote server, exposing its object interface to clients on different machines (threaded server) is

performed by the system administrator. And the server needs to register the object class ID with the remote system.

3 Development Team

3.1 Key Contractor Participants

Karen Smith, PI Coney Island Incorporated . (718) 286 -7873

Steve Cohen, co- PI, Coney Island Incorporated (718) 286-8457.

Larry Doe, Business Official Coney Island Incorporated (718) 286-4444.

Karen Jones, Scientist, Coney Island Incorporated (718) 286-8547.

Roger Mexcur, Scientist, Coney Island Incorporated (718) 286-4444

Karen Nelson, Scientist, Coney Island Incorporated (718) 286-8547

3.2 Key NASA Participants

Jane Black, COTR GSFC (301) 286 -7873

Roger Smith, Contract Specialist GSFC (301) 286-8457.

Ian Green, CSFC TCO (301) 286-4444.

3.3 NASA and Non-NASA Advisors

Jane Black, Boeing Corporation (301) 286 -7873

Roger Smith, Boeing Corporation (301) 286-8457.

Ian Green, Space Science Corporation (301) 286-4444.

Brett Jones, Space Science Corporation (SSTO) (301) 286-8547.

Sam Blue University of Maryland Business Corporation (301) 286-4444

Frank Nelson International Business Machines Corporation (301) 286-8547

4 Technical Activities

4.1 Recent Technical Activities

The term Quality of Service (QoS) is used when addressing such network based technologies and implementations used in the maximization and management of bandwidth resources.

Two main categories make up QoS they are “Resource Reservation” and “Prioritization”. Reservation based implementations act as resource managers while prioritization services ensure that the highest priority applications are granted access over less vital applications. Resource Reservation implementations distribute the bandwidth according to established network policy while “Prioritization” ensures that network hierarchy policies are abided.

The QoS, OPC network architecture to be implemented will maximize interoperability among components by implementing QoS solutions that are open-ended and allow for the testing of new and better technologies. OPC network QoS implementations are to address miscellaneous environment needs with different levels of safety and real time performance. This fact further bounds QoS implementations to the integration of criteria that will meet individual environment needs. Such a provision decrees QoS initiatives to be heterogeneous in nature.

QoS requirements are directly tied to the OPC network hierarchy. Given nodes may need to assume an idle stage when applications performing critical data transfers require additional resources. Such QoS requirements are unique to every OPC network therefore flexibility must be built into the OPC server and client to accommodate applications with special requirements.

Enhancing data throughput rates can also be achieved by transferring information at predefined rates. This feature can be utilized in exercising QoS measures as long as algorithmic flexibility accounts for unique events that may deviate from such uniformity.

Beyond the hierarchy considerations of a network, the efficiency of data transfer implementations strategy must be accompanied by data integrity maintenance implementations responsible for accounting for distorted data, data loss, data transfer delays etc.

Although flexibility is of highest priority when implementing OPC networks, flexibility must be counterbalanced by the implementation of a high degree of deterministic behavior. Such a behavior will be in part defined by the concurrent repetitive tasks of the network.

The OPC Data Custom Interface Specification defines data access within an OPC implementation as being bounded to the item tag connections that make up the OPC group objects. Individual items are not accessible outside the OPC group object. Items have unique 'Value', 'Quality' and 'Time Stamp' characteristics.

The average OPC implementation is responsible for collecting data from physical devices. Data is then distributed to clients such as Supervisory Control and Data Acquisition (SCADA) or Distributive Control Systems (DCS). The network relationships that exist within OPC networks are atypical when compared with standard networks. OPC networks are often made up of multiple servers and clients. That is in a LAN, multiple sensors are physically tied to one computer acting as the server. The client collects sensory data from and may perform further calculations and tasks based on such information (See **Error! Reference source not found.**).

Implementing QoS measures within OPC networks must take into account the unique architecture of the network and exploit possible server and client side implementations.

4.2 Future Technical Activities

On the server side, strategies to be used in the improvement of the server's performance will include studies focusing on understanding server behavior under various loads.

Tasks are to be performed at the server side wherever possible and data collected is to be delivered by sending small data packets to the client whenever possible. Server side processing will also aid in reducing the amount of bandwidth utilized. Caching servers can be deployed within the OPC network to cache frequently used elements. Caching servers when strategically placed, as close as possible to client applications, minimize the path data has to travel and accelerate data transfer speeds. Caching will also help to reduce latency and alleviate bottleneck. One additional feature of caching performance is that it is closely tied to the number of nodes that makeup the network. Increasing the number of clients that makeup the network will in fact improve performance since the probability that a cached document is recalled increases. One disadvantage of Caching is that it mainly aids with the transfer of static information and may not alleviate congestion in cases where only dynamic data is being exchanged.

5 Commercialization Activities

5.1 Recent NASA Potential Customer Activities

Earth Science Technology Office (ESTO). The ESTO applications are for grants and contacts management. The principal contacts are Karen Moe (301) 286 -7873 and Steve Smith (301) 286-8457.

Space Science Technology Office (SSTO). The SSTO applications are for grants and contacts management. The principal contacts are John Doe (301) 286-4444 and Steve Jones (301) 286-8547.

Small Business Innovations Research (SBIR). The SBIR applications are for grants and contacts management. The principal contacts are Paul Mexcur (301) 286-4444 and Robert Nelson (301) 286-8547.

5.2 Recent Non-NASA Potential Customer Activities

Earth Science Technology Office (ESTO). The ESTO applications are for grants and contacts management. The principal contacts are Karen Moe (301) 286 -7873 and Steve Smith (301) 286-8457.

Space Science Technology Office (SSTO). The SSTO applications are for grants and contacts management. The principal contacts are John Doe (301) 286-4444 and Steve Jones (301) 286-8547.

Small Business Innovations Research (SBIR). The SBIR applications are for grants and contacts management. The principal contacts are Paul Mexcur (301) 286-4444 and Robert Nelson (301) 286-8547.

5.3 Other Recent Commercialization Activities

Small Business Innovations Research (SBIR). The SBIR applications are for grants and contacts management. The principal contacts are Paul Mexcur (301) 286-4444 and Robert Nelson (301) 286-8547.

Small Business Innovations Research (SBIR). The SBIR applications are for grants and contacts management. The principal contacts are Paul Mexcur (301) 286-4444 and Robert Nelson (301) 286-8547.

5.4 Future Commercialization Activities

The OPC Data Custom Interface Specification defines data access within an OPC implementation as being bounded to the item tag connections that make up the OPC group objects. Individual items are not accessible outside the OPC group object. Items have unique ‘Value’, ‘Quality’ and ‘Time Stamp’ characteristics.

6 Resources Status

The total cumulative costs incurred, as of 12/31/01, is 45,398.86 dollars. Up to the report date, an estimated 34% percentage of the work has been completed. The details of resource status are given in Table 1.

Table 1. Resource Plan

	Network Implementations	
	P.O.P. 11/02/2001-5/03/2002	
	Contract #NAS13-02001	
	Current Report Period	TOTAL C-T-D
DIRECT LABOR	9,470.93	9,470.93
DIRECT TRAVEL	514.04	514.04
DIRECT CONSULTANT	-	-
DIRECT MATERIALS	-	-
TOTAL DIRECT COSTS	9,984.97	9,984.97
OVERHEAD		

99.80%	9,451.99	9,451.99
GENERAL & ADMIN		
20.90%	4,062.32	4,062.32
TOTAL COSTS	23,499.28	23,499.28
Estimate of Cost to Complete the Contract		45,398.86
Estimated Percentage of Physical Completion of the Contract		34%

7. References

Brockschmidt , Kraig, Inside OLE, Second Edition, Microsoft Press, Redmond, WA, 1995.

Microsoft COM Specification, version 0.9, 10/24/95 (available from Microsoft's FTP site).

Microsoft Systems Journal, Q&A, April, 1996, pp. 89-101.

OLE Automation Programming Reference, Microsoft Press, Redmond, WA, 1996.

OLE 2 Programming Reference, Vol. 1, Microsoft Press, Redmond, WA, 1994.

OLE for Process Control: Data Access Custom Interface Standard, Version 2.04. Sept.5, 2000

OLE for Process Control: OPC Overview, Version 1.0, Oct. 27, 1998