Implementing a Grants.Gov System-to-System Interface using the Microsoft .NET Framework

Summary

The National Endowment for the Humanities has developed a Grants.Gov system-to-system interface that may serve as a low-cost model for other agencies using the Microsoft .NET Framework (.NET). NEH is publishing this white paper in the hopes that it will assist other agencies in their efforts to participate in Grants.Gov. If you have any questions or comments about the materials in this paper, please don't hesitate to contact us. Also, to view the latest version of this document (and to see sample code), please see our website at: http://www.neh.gov/whoweare/cio.htm

Contacts

Questions or comments may be directed to:

Beth Stewart, Information Technology Specialist
National Endowment for the Humanities
bstewart@neh.gov

Brett Bobley, Chief Information Officer
National Endowment for the Humanities
bbobley@neh.gov

website: http://www.neh.gov/whoweare/cio.htm

Background Information

The National Endowment for the Humanities (NEH) is a small grant-making agency that receives approximately 5,000 grant applications annually from both individual and institutional applicants. With a small number of technical and grant program staff members and a limited Information Technology budget, NEH decided to develop in-house a Grants.Gov system-to-system interface using existing servers and applications known by the agency's IT staff members. A system-to-system interface promised receipt of grant applications submitted via Grants.Gov with minimal oversight and involvement, an important requirement since the agency could not add staff members for the handling of these applications.

With these goals in mind, the system-to-system interface and downloaded applications would need to be developed and accessed with the following existing tools and utilities:

- Microsoft Visual Studio .NET 2003 with the Visual Basic .NET language
- Microsoft Web Services Enhancements (WSE) 2.0
- Microsoft SQL Server 2000
- Microsoft Internet Information Services (IIS)
- Microsoft .NET Framework 1.1
- Microsoft Internet Explorer
- Adobe Acrobat Reader

Though NEH IT staff members recognized the relative challenge of developing a .NET system-to-system interface instead of a Java-based interface, the project was pursued because it could be developed with existing tools and managed by system administrators trained in Microsoft technologies.

During development, the following additional low-cost APIs were obtained:

- DynamicPDF Merger for .NET by ceTe software, used to merge application files into a single, printer-friendly application file in PDF format, and
- SharpZipLib by ic#code, an open source ZIP library for .NET.

The NEH system-to-system interface entered its production phase with two pilot grant programs in October 2004. After the conclusion of successful testing, in 2005 all NEH institutional grant programs have been advertised on Grants.Gov. In October 2005, the interface was upgraded to take advantage of the new Grants.Gov support for DIME attachments. To achieve our goal of minimal staff involvement, the interface automatically downloads applications every hour that have a status of "Validated." As applications are downloaded, they become available immediately to staff members who review applications for eligibility and are added to the NEH Grants Management System database.

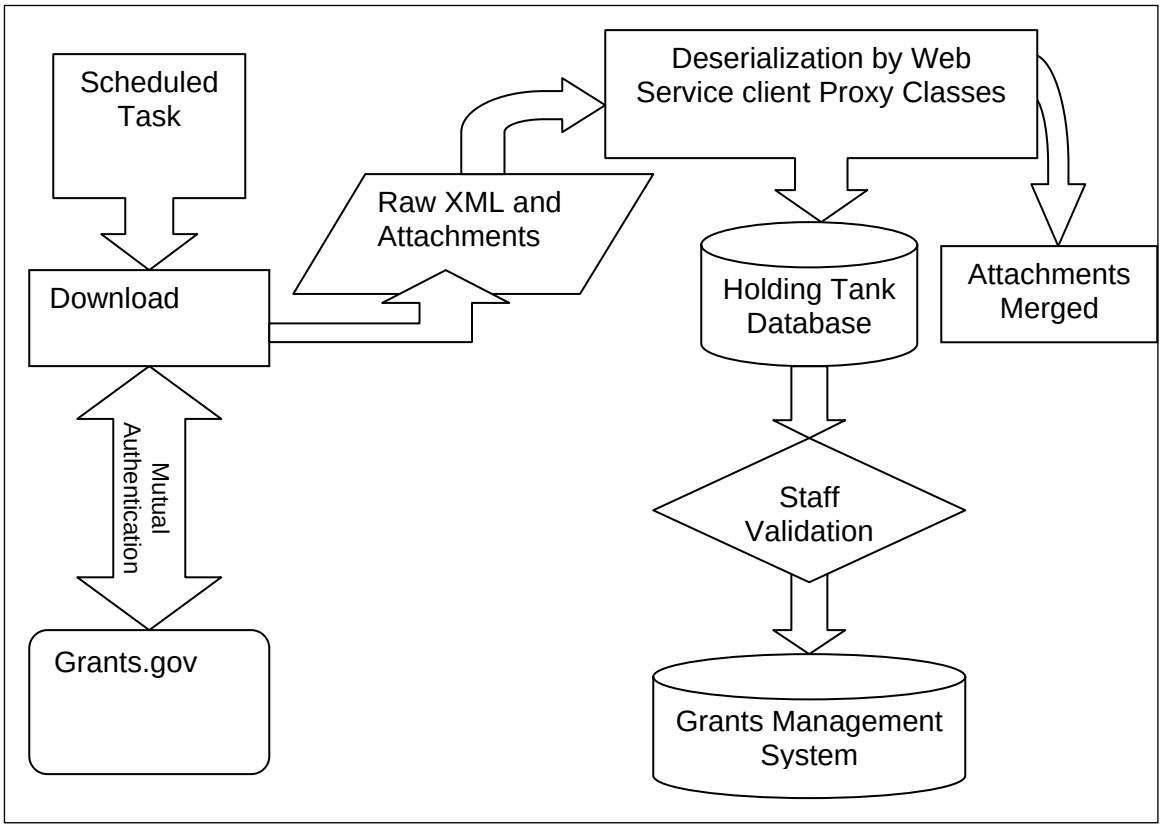
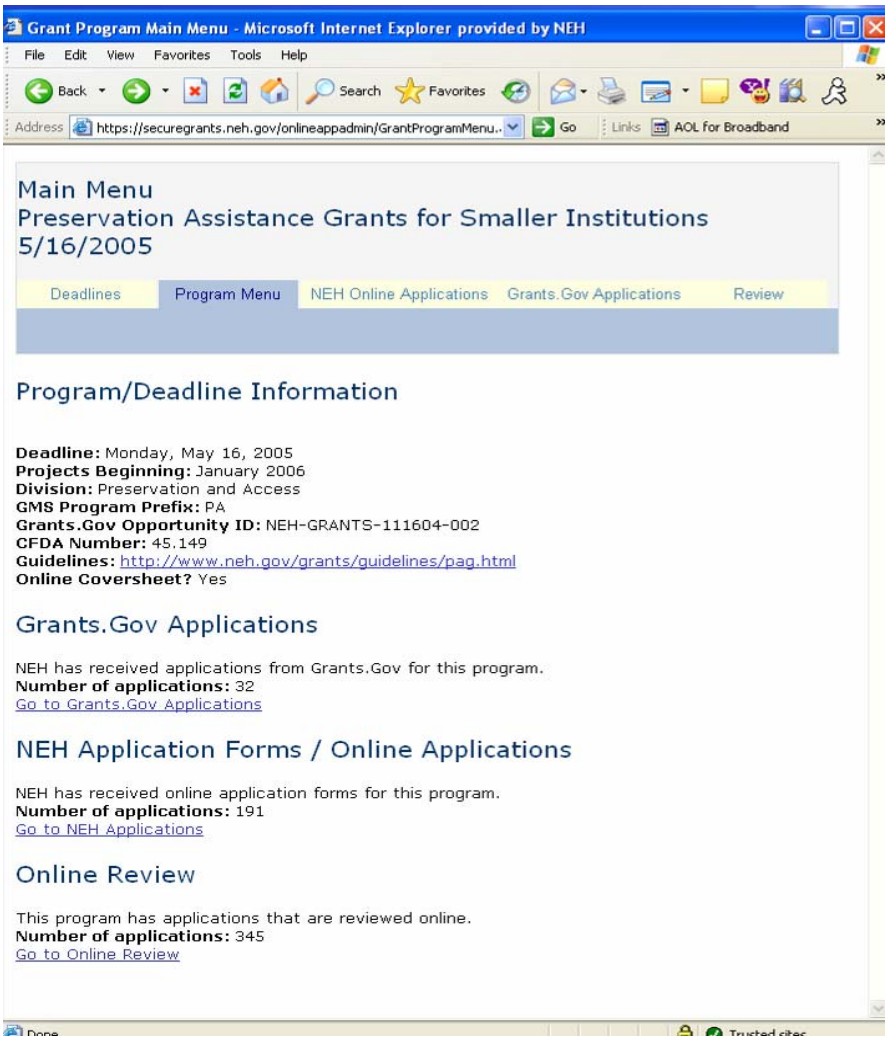


Figure 1. Logical Description of the NEH system-to-system interface

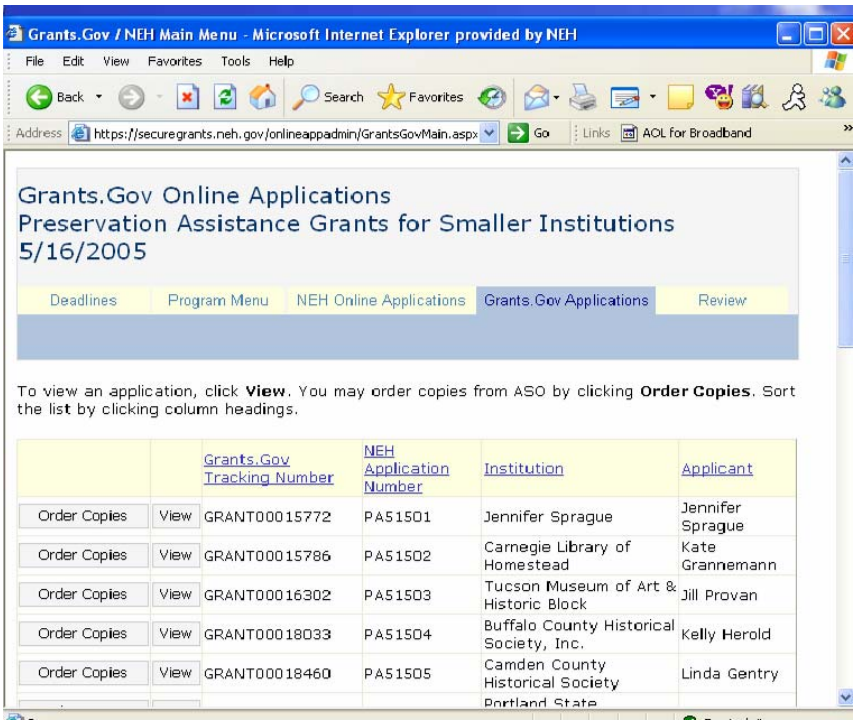
Integration with Existing Systems

A web-based application system has been in place at NEH since 2002 for the receipt, review, and management of applications from individual applicants, whose numbers total more than half of the applications received by NEH. Because the existing online application system has been widely adopted by NEH grant programs, Grants.Gov applications were tightly integrated with the existing system to encourage adoption and promotion of Grants.Gov as a viable and staff-friendly alternative to NEH online application forms.

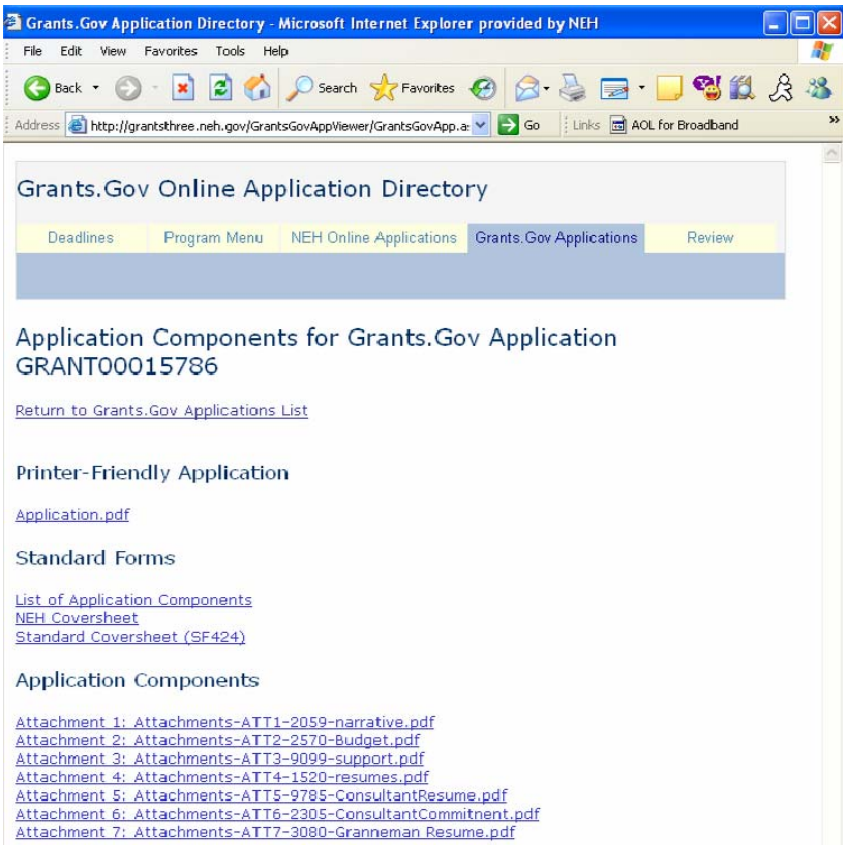
A single grant program accessed online offers the following information to staff members:



By following the "Go to Grants.Gov Applications" link, the list of applications that were downloaded via the system-to-system interface is displayed:



A single application consists of a series of separate files and a single file that merges the application components into a printer-friendly PDF format. This single file simplifies printing and offers a way to send Grants.Gov applications to reviewers.



Web-based access to Grants.Gov applications has minimized the involvement of IT staff members in providing copies of applications, as grant programs are able to order copies and even save locally a grant application.

Technical Details: Instructions for Building a .NET System-to-System Interface

Development of a .NET system-to-system interface is less documented than one developed using Java, particularly because a .NET reference implementation does not exist. Below, selected technical details of the solution adopted by NEH are described as a series of steps to develop a simple system-to-system interface that queries Grants.Gov for available applications and downloads these applications in .zip format.

Step 1: Microsoft Web Services Enhancements

In October 2005, Grants.Gov began to support the Direct Internet Message Encapsulation (DIME) message encapsulation format, which is used by the Microsoft .NET platform to exchange attachments with SOAP messages. Though Grants.Gov continues to support attachments using either DIME or multipart MIME, because .NET provides no support for multipart MIME the simpler choice is to develop a solution using DIME. To use the DIME protocol for handling attachments in a SOAP message, Microsoft Web Services Enhancements (WSE) 2.0 must be installed on all servers that will run the system-to-system interface. As of November 2005, the most recent version of WSE is available for download here:

http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en

After WSE is installed, developers will also gain access to WSE documentation.

Step 2: Create a New Project and Reference the Grants.Gov Web Service

Within Microsoft Visual Studio, choose to create a new project that is a console application in the language of your choice. Sample code is provided in Visual Basic .NET. Once the project has been created, an additional step is needed to use Web Services Enhancements: right-click the project name and select "WSE Settings 2.0." Under the General tab, check the box to Enable the Project for Web Services Enhancements and click Ok.

Next, a web reference must be added to the Grants.Gov web service. To add a web reference, right-click the project name and select "Add Web Reference." Enter http://atws.grants.gov/wsdl/agency/AgencyIntegrationServices-V1.2.wsdl as the URL and click "Go" to find the web service. When the web service has been found, click the "Add Reference" button.

Once the web reference has been added, it will appear in the Visual Studio Solution Explorer as gov.grants.atws. Before continuing, check that the web reference is enabled for WSE by right-clicking the gov.grants.atws web reference and choosing View in Object Browser. Expand the gov.grants.atws namespace and check that a class exists named AgencyIntegrationServicesWse. If not, confirm that the project has been enabled for web services enhancements (as described above), right-click the web reference, and select Update Web Reference.

The use of dynamic URL behavior will significantly simplify use of the interface as it will allow a developer to change the URL of the web service (e.g. from the test

atws.grants.gov to the production ws.grants.gov server) from a configuration file. Right-click the Grants.Gov web reference, which by default is named gov.grants.atws, and select Properties. Within the properties view, change the URL behavior from Static to Dynamic. When the URL behavior is changed to Dynamic, the app.config file is altered automatically to include an appSettings element as below:

```
<appSettings>
<add key= "GrantsGovSampleDIMEClient.gov.grants.atws.AgencyIntegrationServices"
    value="https://ws.grants.gov:446/agency-s2s-
server/services/AgencyIntegrationSoapPort"/>
</appSettings>
```

When running the application, the URL in the value attribute will control the server used by the web service client.

Step 3: Install and Use a Certificate for SSL

To prepare for integration testing with Grants.Gov, a server certificate must be created and signed by a Certificate Authority such as VeriSign or Entrust. Once the certificate is installed on the server that will run the Grants.Gov system-to-system interface and has been installed by Grants.Gov, use the certificate export tool within IIS to export the certificate (without private keys) in DER encoded binary X.509 format, which will produce a .cer file. This file will be referenced within the system-to-system interface after the web refererence is instantiated:

```
    ' Instantiate the Web Service
    Dim gg As New gov.grants.atws.AgencyIntegrationServicesWse

    ' Associate a certificate with the web service
    Dim x509 As X509Certificate = X509Certificate.CreateFromCertFile _
       ("c:\MyCertificate\myCert.cer")
    gg.ClientCertificates.Add(x509)
```

Step 4: Use WSE Functionality to Access Attachments

For this simple client, we will execute the GetApplicationList method to obtain a list of applications available for download. Enumerating through the list of available applications will cause execution of the GetApplicationZip method for each application. (For complete code, please see the provided .zip archive.) WSE-specific features of SOAP messages are used to access the attachment containing the application .zip archive, which is available within the ResponseSoapContext:

```
   ' Prepare the request for the zip file
   Dim getZipReq As New gov.grants.atws.GetApplicationZipRequest
   Dim getZipRes As gov.grants.atws.GetApplicationZipResponse
   getZipReq.Grants_govTrackingNumber = "GRANT000xxx"

   ' Request attachments sent using DIME protocol
   getZipReq.encodingType = gov.grants.atws.EncodingTypeEnum.DIME

   ' Execute the web service call
    getZipRes = gg.GetApplicationZip(getZipReq)

    ' Save the zip file, which is stored in the current web service context
    Dim respContext As SoapContext = gg.ResponseSoapContext
    Dim zipFile As New FileStream("GRANT000xxx.zip", FileMode.Create, _
        FileAccess.Write)

    Dim readBuffer(4096) As Byte
```

```
Dim bytesread As Integer = 0
Do
  bytesread = respContext.Attachments(0).Stream.Read(readBuffer, 0, _
              readBuffer.Length)
  zipFile.Write(readBuffer, 0, readBuffer.Length)
Loop While bytesread > 0
```

Conclusion

Though a Grants.Gov system-to-system interface using the Microsoft .NET framework is less documented, it is relatively simple to implement. The functionality of the simple client described above may be extended to handle deserialization of xml files, merging of application components into a single PDF document, and database storage.

References

Balena, Francesco. *Programming Microsoft Visual Basic .NET*. Microsoft Press, 2002.

Bosworth, Adam, et. al. "XML, SOAP, and Binary Data." http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/infoset_whitepaper.asp.

Powell, Mat. "Programming with Web Services Enhancements 2.0". http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwse/html/programwse2.asp.