



NYS OFFICE OF GENERAL SERVICES

Serving New York

Division of Financial Administration

ADDENDUM No. 1

Request for Quotes (RFQ) No. 1768

Date: July 25, 2012

Subject: RFQ Clarifications, Questions and Answers

Title: IT Services to Develop a Web-Based Application "eHBITS" for the Hourly-Based IT Services (HBITS) Contract

Group: 73012

Address Bid Inquiries to: Kathleen Gallagher
Purchasing Agent
NYS Office of General Services
40th Floor, Corning Tower
Empire State Plaza
Albany, NY 12242
Email: Kathleen.Gallagher@ogs.ny.gov

To Prospective Bidders: This addendum is to provide the official clarifications and answers to written questions submitted by prospective bidders regarding RFQ #1768.

All proposals are due to OGS by Thursday, August 2, 2012, at 2 PM EST.

Clarifications, Questions and Answers:

Question #1:

RFQ Section 1.A - In the executive summary of the RFP it states: "The Office of General Services (OGS) is soliciting quotes from New York State small businesses and certified Minority / Women owned Business Enterprises (M/WBEs) to develop a web-based application to implement the solutions identified in the Hourly-Based IT Services (HBITS) RFP."

I wanted to verify that out of state contractors are excluded based on this qualification requirement. Is this correct?

Request for Quotes (RFQ) No. 1768

ADDENDUM No. 1

Answer #1:

Out-of-state companies that have received New York State certification as a Minority / Women-Owned Business Enterprise (MWBE) may participate in this RFQ.

Question #2:

RFQ Section 1.A - We are an approved Minority Women Owned Business with the UN in NYC do we need to have a separate approval/certification for this FRP and if YES what is the process to apply.

Answer #2:

Companies must have New York State certification as a Minority / Women-Owned Business Enterprise (MWBE). Empire State Development is the agency that administers New York State's MWBE certification program. Please visit their website at www.esd.ny.gov to obtain information on eligibility and how to become certified.

Question #3:

RFQ Section 1.A - Please reply to let me know if my company is eligible to participate in RFQ #1768. I have been a lifelong NY State resident but my business is located in CT. We are certified by the CT Department of Administrative Services as Small Business Enterprise.

Answer #3:

This RFQ is only available to vendors that are either a New York State small business or a New York State certified Minority / Women-Owned Business Enterprise (MWBE). For a company to be considered a New York State small business (SBE), they must meet the following:

- (a) business is located in New York State;
- (b) business is independently owned and operated;
- (c) business is not dominant in its field; and,
- (d) business employs one hundred or fewer persons.

Question #4:

RFQ Section 1.A - In the executive summary on page 3, it says: "The Office of General Services (OGS) is soliciting quotes from New York State small businesses and certified Minority / Women owned Business Enterprises (M/WBEs) to develop a web-based application to implement the solutions identified in the Hourly-Based IT Services (HBITS) RFP." Are you only accepting quotes from small businesses located in New York? Or can it be a business located in another state that is licensed to do business in New York State?

Answer #4:

Please see the answer to question #3.

Question #5:

RFQ Section 1.A - We are based out of Phoenix, AZ. Can we bid on this opportunity? Of course, we are registered and we got this bid in our mail box. But as such can we bid on this opportunity.

Answer #5:

Please see the answer to question #3.

Question #6:

RFQ Section 1.A - Can we bid on all the positions or do we have to only bid on 4 positions? I see that only 4 individuals need to be submitted by each Contractor. Can you clarify on this?

Answer #6:

There are not four positions to bid on as this is not a staff augmentation project. OGS is selecting a proposal in its entirety to perform the services in Section 3 of the RFQ. OGS has instituted a condition where a prospective bidder may not propose more than four candidates as part of its response to this RFQ.

Question #7:

RFQ Section 1.A - On RFQ Number 1768: IT Services to Develop a Web-Based Application “eHBITS” for the Hourly- Based IT Services (HBITS) Contract, on page 3 it states “small businesses and certified Minority / Women owned Business Enterprises (M/WBEs).” Does this mean a small business can bid, or does it mean that the bid must come from a small business and one that is certified M/WBE?

Answer #7:

Please see the answer to question #3.

Question #8:

RFQ Section 1.A - Executive Summary – 2nd paragraph – “...*The Office of General Services (OGS) is soliciting quotes from New York State small businesses and certified Minority / Women owned Business Enterprises (M/WBEs)....*” Does this mean a small business enterprise who is not a WMBE is eligible to submit a bid? And if yes, does NYS OGS certify small business enterprises?

Answer #8:

Please see the answer to question #3.

Question #9:

RFQ Section 1.A - Is this RFQ open only to NYS small businesses and MWBE organizations?

Answer #9:

Please see the answer to question #3.

Question #10:

RFQ Section 1.A - Are only New York State small businesses and certified Minority / Women owned Business Enterprises (M/WBEs) allowed to bid on this RFQ?

Answer #10:

Please see the answer to question #3.

Question #11:

RFQ Section 2.B - Go Live Date in November 1 2012. How can that be if the OSC Approval is September 1st, 2012?

Answer #11:

OGS is seeking a Bidder who can deliver a solution in two months. OGS understands it is seeking an aggressive deadline for this project but is confident the solution can be developed in the timeframe prescribed in the RFQ.

Question #12:

RFQ Section 2.B - There is only about a week between OGS supplying answers to questions and the due date for proposals. Because the answers to our questions may determine if and how we formulate our proposal, we consider one week to be a tight schedule. Is there any chance of OGS accelerating their answers or extending the proposal submission deadline?

Answer #12:

Please see the answer to question #11.

Question #13:

RFQ Section 2.B - By what date do you expect the winning bidder will be able to start work? If this date is postponed through no fault of the bidders (e.g., because of delay in selecting bidder), will the GO LIVE date also be postponed?

Answer #13:

OGS cannot provide a specific start date as such date is contingent on the approval of the resulting contract by the Office of the State Comptroller (OSC). Also, see answer to question #14.

Question #14:

RFQ Section 2.B - If the OSC approval of the contract were to slip (go beyond the estimated September 1, 2012 date) then would the System Go Live date be extended?

Answer #14:

Yes.

Question #15:

RFQ Section 2.C - This states that this is a three-month project but there is a provision to extend two additional months for a total of five months and from the Contract Start to the System Go Live is only two months. What is the project duration and is there a penalty for missing the November 1, 2012 date?

Answer #15:

The project duration is 3 months. Delays in the go-live date that are caused by New York State will not result in a penalty to the vendor.

Question #16:

RFQ Section 2.D - Is there a maximum number of persons that can work on this project (i.e., if work must be performed at your offices in Albany, how many seats are available).

Answer #16:

Bidders may only bid a maximum of four people for this effort. However, additional employees or subcontractors may work on the solution if no cost is passed along to OGS.

Question #17:

RFQ Section 3.A - Is this a project-based or staff-augmentation RFQ? In other words, will a single winning bidder be filling all the roles outlined on pages 5 and 6, or might OGS select certain candidates from one bidder and other candidates from another?

Answer #17:

This is a project-based RFQ in which only one Bidder will be selected. Also, please note that OGS is not seeking a Bidder to provide staff for all job titles listed in Section 2.D.1 of the RFQ.

Question #18:

RFQ Section 3.A - We reviewed this bid and read through the RFQ. Our company can build custom applications but are you aware that there are many off the shelf solutions that exist in the marketplace like Consol <http://www.consolvms.com/default.aspx>

We wanted to better understand if OGS has decided to build your own tool versus using something off the shelf or if you may not have been aware that there is a whole industry that exists that is focused on selling what is called VMS solutions that perform many of the functions in your RFQ.

Our company uses Consol at Pfizer to manage all of their contingent workforce.

We look forward to receiving your feedback.

Answer #18:

OGS is aware of industry applications but has decided to build its own tool.

Question #19:

RFQ Section 3.A - Is there a preferred architecture for the proposed system?

Answer #19:

There is no preferred architecture for the proposed system other than that defined in the RFQ and in these answers to questions submitted by interested bidders.

Question #20:

RFQ Section 3.A - Can you supply to us the OGS Application Standards Documentation?

Answer #20:

The OGS coding standards document is attached.

Web-based information and applications shall be compliant with certain accessibility standards (noted in the table below) developed by the Federal government in compliance with Section 508 of the Rehabilitation Act of 1973, as amended 29 U.S.C. § 794 (d).

Section 508 Standards

Subpart B, section 1194.22 Web-based Intranet and Internet Information and Applications

Subpart C, section 1194.31 Functional Performance Criteria

Question #21:

RFQ Section 3.A - Can you supply to us the OGS Application Security Standards Documentation?

Answer #21:

OGS does not have a reference document documenting specific security standards. All applications are expected to meet industry standard security practices.

Question #22:

RFQ Section 3.A - Can you elaborate on your external security testing requirements?

Answer #22:

Prior to deploying the application, OGS will use standard vulnerability testing tools to ensure that the application does not contain security holes (such as SQL injection, cross-site scripting).

Question #23:

RFQ Section 3.A - What browsers do you expect the system to work on? Do you expect it to be compatible with mobile devices?

Answer #23:

The application should be coded to support Internet Explorer, Chrome, and Firefox. Safari and Opera are optional, if possible. OGS does not expect the solution to be compatible with mobile devices at this time.

Question #24:

RFQ Section 3.A - Are you open to working under an Agile project management structure?

Answer #24:

Yes.

Question #25:

RFQ Section 3.A - What is the possibility of the eHBITS contract being extended beyond the current scope with additional funding?

Answer #25:

No possibility as this procurement is capped at a not to exceed amount based on State Finance Law.

Question #26:

RFQ Section 3.A - Are there any specific database or authentication systems that the proposed eHBITS application must be coded to interface with?

Answer #26:

The NYS Office of Information Technology Services hosts a Directory Services application called NYSDS that runs on SiteMinder. It is anticipated that this application will interface with the NYSDS application for authentication. OGS will be responsible for establishing the communications between the SiteMinder server and the eHBITS application server.

Question #27:

RFQ Section 3.A - What is the current or proposed operational environment for the eHBITS Application?

- a. hardware (i.e., PC servers running Microsoft Windows Server)
- b. operating system (i.e, Microsoft Windows)
- c. web server (i.e., Microsoft IIS)
- d. database (i.e, Microsoft SQL/Server)
- e. development software (i.e., Microsoft Visual Studio)

Answer #27:

eHBITS will run on VMWare virtual servers running Windows 2008 R2 server and IIS 7.0. The SharePoint version is 2010. The database will be Microsoft SQL 2010. The development platform used at OGS is Visual Studio 2008.

Question #28:

RFQ Section 3.A - States – “The selected vendor, working with OGS Information Resource Management (IRM) staff, will develop a web portal solution/application with a MS Office solution for the back-end database, as described in Exhibit 2, to be utilized by both OGS and Authorized Users of the HBITS contracts. The MSP will have the ability to see manage the HBITS process through a working MS Office system. IRM will establish a development area in which the vendor will be able to code the application. The responsibility to migrate any part of the application to the test and production platforms will fall to IRM. **The winning vendor is expected to perform their work at the office of OGS IRM, located on the 36th floor of the Corning Tower, Albany, NY.**” Is there an option to do some or most of the development off-site? If not why (is there a technical reason)?

Answer #28:

Yes, there will be an option to do some development off-site. OGS will require weekly status reports on a pre-determined date but staff will have to be on-site for portions of the project.

Given that there are very tight timeframes under which this project must be completed, there is

going to have to be close collaboration between the winner bidder and OGS. Such collaboration is best done in person. Performing 100% of the work at the OGS offices is not a strict requirement, but if a vendor is proposing to perform some work off-site, the proposal should clearly define what part of the work will be performed on-site and which will be performed off-site. The quote must also contain a description of how work hours will be tracked and reported to OGS by deliverable and how OGS will be able to review the work product during it development. Vendors also must ensure that their development platform is consistent with that of OGS.

Question #29:

RFQ Section 3.A - Must all team members work at the Albany location, or could some of the work be performed at the vendor's site?

Answer #29:

See Answer #28.

Question #30:

RFQ Section 3.A - Will your office produce the development environment, or must the consultant team supply their own laptops and development software?

Answer #30:

OGS will provide PCs and the required development software.

Question #31:

RFQ Section 3.A - What is the back-end database environment?

Answer #31:

Microsoft SQL 2010.

Question #32:

RFQ Section 3.A - On page 11, it says, "Throughout the eHBITS process, the MSP will have the ability to manage, monitor and record data through a working Microsoft Office database." What is meant by a "Microsoft Office database"?

Answer #32:

The intent of this statement is that the eHBITS application will allow the MSP to monitor and update the status of the work request throughout its life cycle through the data stored in the MS SQL database.

Question #33:

RFQ Section 3.A - Will we be required to work on site exclusively and full-time, or can we complete some or most of the work in our own offices?

Answer #33:

Given that there are very tight timeframes under which this project must be completed, there is going to have to be close collaboration between the winner bidder and OGS. Such collaboration is best done in person. Performing 100% of the work at the OGS offices is not a strict requirement, but if a vendor is proposing to perform some work off-site, the proposal should clearly define what part of the work will be performed on-site and which will be performed off-site. The quote must also contain a description of how work hours will be tracked and reported to OGS by deliverable and how OGS will be able to review the work product during its development. Vendors also must ensure that their development platform is consistent with that of OGS.

Question #34:

RFQ Section 3A - Is OGS open to taking advantage of open source software libraries built for the .NET Framework to lower project costs and enhance product features?

Answer #34:

Yes.

Question #35:

RFQ Section 3.A - Is OGS open to taking advantage of a streamlined and optimized off-site Agile project management environment to help reduce cost?

Answer #35:

Yes.

Question #36:

RFQ Section 3.A - How does OGS envision using SharePoint or InfoPath? Is OGS looking to utilize those types products as a CMS?

Answer #36:

OGS envisions the SharePoint platform to be leveraged for its ability to manage data and workflow. Infopath is one mechanism for simplifying the data entry process into eHBITS. The goal is to be able to use these tools to simplify the development process for this application.

Question #37:

RFQ Section 3.A – “OGS is seeking to automate this process by leveraging MS Office products, most notably SharePoint and InfoPath, to achieve greater efficiency.” Do you have any other applications at OGS that are using SharePoint and InfoPath and are in production?

Answer #37:

OGS is currently using SharePoint 2010 for simple document management, list processing, and collaboration. We have a few simple instances where Infopath is used for data submission into SharePoint.

Question #38:

RFQ Section 3.A - Can OGS provide current version of office products; specifically InfoPath and SharePoint? Is it safe to assume that OGS has sufficient development and production licenses for the same?

Answer #38:

OGS is running SharePoint 2010. All OGS staff who will utilize this application will be upgraded to Office 2010 before the implementation of the application. OGS will handle any license needs.

Question #39:

RFQ Section 3.A - As per #4 under Scope of Work (RFQ page 12), the proposed web-portal solution must incorporate a MS Office product, such as SharePoint and InfoPath. Is OGS willing to consider a web-portal solution which is based on Microsoft technology platform such as .NET and SQL Server but not on SharePoint and InfoPath?

Answer #39:

Yes.

Question #40:

RFQ Section 3.A - As per the RFQ, developer personnel are expected to work at the OGS facility. Is OGS willing to consider if development work is carried out at vendor's facility and vendor's project manager is available at OGS as and when needed (for example: at the time of requirements gathering, status meetings, handover, implementation etc.)?

Answer #40:

Given that there are very tight timeframes under which this project must be completed, there is going to have to be close collaboration between the winner bidder and OGS. Such collaboration is best done in person. Performing 100% of the work at the OGS offices is not a strict requirement, but if a vendor is proposing to perform some work off-site, the proposal should clearly define what part of the work will be performed on-site and which will be performed off-site. The quote must also contain a description of how work hours will be tracked and reported to OGS by deliverable and how OGS will be able to review the work product during it development. Vendors also must ensure that their development platform is consistent with that of OGS.

Question #41:

RFQ Section 3.A - What kind of hosting environment is available or provided by OGS? Is the vendor responsible for deploying or supporting the deployment of eHBITS on State's servers?

Answer #41:

OGS will provide a test, staging and production set of platforms for the application. The vendor will work with the appropriate OGS IRM staff to deploy the application.

Question #42:

RFQ Section 3.A - Is there any post-deployment support expected from vendor?

Answer #42:

Yes, but it would only occur within the contract term.

Question #43:

RFQ Section 3.A - Is there any user-training expected as part of the scope?

Answer #43:

User training will only occur with the State's project manager and other members of the HBITS team. At this time, the State does not envision this training to exceed 10 people.

Question #44:

RFQ Section 3.C - Can a M/WBE firm awarded this contract act as a subcontractor to an approved vendor to provide IT services under HBITS contracts?

Answer #44:

Yes, as long as the M/WBE firm is not one of the 25 vendors previously awarded an HBITS contract.

Question #45:

RFQ Section 3.C - Can one of 25 vendors awarded the HBITS contract act as a subcontractor to a M/WBE firm to propose a combined solution for this project?

Answer #45:

No. The 25 HBITS vendors may not act as either a prime or a subcontractor for any proposal submitted in response to RFQ 1768.

Question #46:

RFQ Section 3.C - If we were not an approved Vendors from the HBITS award outcome, are we able to bid on these procurements on the website?

Answer #46:

If this question refers to the HBITS contracts, then the answer is no. Only those 25 HBITS vendors are ineligible to bid on this RFQ.

Question #47:

RFQ Section 3.C - Page 14.C - Out-of-Scope Work - The second bullet indicates that the awarded vendor cannot be one of the 25 awarded HBITS vendors. Is this correct and would this change if and when HBITS gets rebid? Please provide specific timeframes.

Answer #47:

Yes, this is correct. The HBITS contracts are five-year awards, thus OGS cannot speculate as to future decisions. For the purposes of a response to this RFQ, only those 25 HBITS vendors are ineligible to bid.

Question #48:

RFQ Section 3.C - Are only the tentative awardees for RFP #22439 allowed to bid on this RFQ?

Answer #48:

Please see the answer to question #45.

Question #49:

RFQ Exhibit 2 - Are the screens presented in the RFP final versions? Are you open to streamlining and modifying screens to optimize user experience?

Answer #49:

No. OGS is open to streamlining and modifying screens to optimize user experience. Bidders should review the OGS public website (ogs.ny.gov) to determine any potential limitations to such optimization.

Question #50:

RFQ Other - Would there be other opportunities such as hardware procurement or systems support attached to this project? Or would the same company that develops the application will be responsible to provide all required support services and products?

Answer #50:

Question 1: No.

Question 2: OGS does not anticipate support services and products as being required for any solution proposed.

Question #51:

RFQ Other - Is the State willing to accept a commercially available software solution (COTS) instead of new development as long as the COTS solution is configured as per State requirements?

Answer #51:

No.

Question #52:

RFQ Other - Has the State considered using a commercially available Vendor Management System (VMS) delivered as a secure cloud service that can save cost and time-to-market compared to custom developed software? Our research has indicated that this option is now being used or considered by a number of local and state governments. This can also provide significant cost savings to the State as all of the funding for configuring, hosting, enhancing and maintaining is borne by a small fee charged to the participating staffing vendors resulting in zero one-time and on-going costs to the State. Lastly, such systems have extensive features including electronic time entry/approval/invoicing, on-demand reporting (standard and ad hoc) and other desirable features which are not currently in the scope of the RFQ but would be extremely useful to the eHBITS users

Answer #52:

Yes, the State has considered commercially available Vendor Management Systems but has decided to go in a different direction. OGS, however, is only seeking a solution for the scope of work outlined in Section 3 of this RFQ.

Intents to Quote were received by 3:00 p.m., Thursday, July 19, 2012, from the following vendors:

Acro Service Corporation
39209 W. Six Mile Rd., Suite 250, Livonia, MI 48152
RV Rao
(734) 542-4318
rvrao@acrocorp.com

Advanced American Technologies, Inc
4700 Rockside Road, Suite # 415, INDEPENDENCE, OH 44131
Ravi Reddy
(440) 339 6260 Mobile / (216) 901 3200 Work
Ravi@advancedameritech.com

Aeon Nexus Corporation
174 Glen Street
Glens Falls, NY 12801
Kiley Wittig
518.338.1551
Kileywittig@aeonnexus.com

Artech Information Systems LLC
240 Cedar Knolls Road
Suite 100
Cedar Knolls, NJ 07927
John Spry - Vice President
973-998-2570, Cell 607-727-6149
nyogs@artechinfo.com

Arthur Brown LLC
16 Kensington Suite 3
Delmar, New York 12054
Arthur M. Brown
518-635-0556 Office
518-265-1205 Mobile
Email: www.arthurbrownllc.com
abrown@arthurbrownllc.com

CMA Consulting Services
700 Troy Schenectady Road
Latham, NY 12110
Jon Barry, Director of Documentation and Marketing
proposals@cma.com / jbarry@cma.com
P 518.429.2256
M 518.461.3233
F 518.783.5093
www.cma.com

Data Industries, Ltd.
1370 Broadway, Suite 519, New York, NY 10018
Paul Raifaizen
212-471-1000
IT70@dataind.com

DatamanUSA, llc
6890 S Tucson Way, Ste 100, Centennial, CO 80112
Gyan Saxena
Work: 720-248-3111
Cell 303-668-4926
GP@DatamanUSA.com

Documentation Strategies, Inc.
15 Second Ave., Rensselaer, NY 12144
Bill Cunningham
(518) 432-1233 ext. 23
Cunningham@docstrats.com

Dyal, Monroe & Salvo, Inc..
1320 East 53rd Street, Brooklyn NY 11234
Amar Dyal
(917) 453-9973
clientservices@dymosal.com

Employee Leasing of Greater New York, LLC
4254 West 33rd Street, Suite 130, New York, NY 10001
Gonzalo Gus Vergara
800-504-1148 Ext 307
gus@distinctivepersonnel.com

GCOM Software Inc,
www.gcomsoft.com
24 Madison Avenue Extn, Suite 4
Albany, NY 12203
Riyaz Ladkhan
Phone (O) 518-869-1671 Extn. 223
518-869-5901 Extn. 223
518-935-2313 Extn. 223
Phone (C) 518-892-1590
Fax(O) 518-869-1673
<http://www.linkedin.com/pub/riyaz-ladkhan/3/429/5b9>

Greene Tree Technology Group, LLC
359 Broadway, Troy, NY 12180
Annmarie Lanese, President
518-331-2831
alanesey@greanetree.com
Jennifer Hixon, Systems Analyst
518-479-0103
jhixon@greanetree.com

ImageWork
170 Hamilton Ave. Suite 301
White Plains, New York 10601
Kevin C. Hansan, President
www.imagework.com
Office: 914.681.0700
Fax: 914.801.2276
Cell: 914.396.8613

Infosys Internationals, inc
110 Terminal Drive, Plainview, NY 11803
Cynthia Silletti
516-576-9494 x 3340
cynthias@infosysinternational.com

InfoTran Systems, Inc
4309 Ligustrum Dr.
Melbourne, FL 32934
Tien Tran
(925) 215-5625
tientran@infotransys.com

Instructional Systems Inc.
401 Hackensack Avenue, Hackensack, NJ 07601
Rosanne Rufo
201-343-3003
rosanne@isinj.com

ITX Corp.
1169 Pittsford- Victor Rd. Bldg. #3, Suite 100
Pittsford, NY 14534
Sylvia (Cadena) Alexander, Sr. Internet Consultant
Tel (office): 800-600-7785 x4827 / 585-899-4827
Tel (cell): 585-202-2185
Tel (eFax): 646-349-2651
Tel (24 x 7 Support): 800-600-7785

Journee Technology Staffing Inc.
2117 Buffalo Road, suite 275 Rochester, NY 14624
Orville Dixon
585-210-5314
journeetech@frontier.com

KMQ Enterprises, Inc. d/b/a Tailwind Associates
1462 Erie Boulevard, Schenectady, New York 12305
René Guzek
518-579-3020
tailwind@tailwindassoc.com

KnowledgePro LLC
Ashwin Kini
609-933-1642 - Tel
609-269-5172 - Fax
ashwin.kini@Knowledgeprollc.com
www.knowledgepro.us

Koryak Consulting, Inc
2003 Kinvara Drive, Pittsburgh PA 15237
Eric Miller
570-236-7479
emiller@koryak.com

Laurus Development Inc.
4747 Southwestern Blvd.
Hamburg NY 14075
Joe Comerate
716-649-1202 ext 1207
joe@lbcbuffalo.com

Operating System Services, Inc. (OSS, Inc.)
481 North Frederick Avenue, Suite 400, Gaithersburg, MD 20877
Denis Goncharuk
301-977-5759, x 18
Denis@ossjobs.com

Project Solvers of America Inc
349 West Commercial Street
Suite 2320
East Rochester, NY 14445
Al Gubiotti, VP
585-764-5663/585-350-5210

Repeat Business Systems
4 Fritz Blvd.
Albany, NY 12205
Dawn Abbuhl, President
Dabbuhl@rbsalbany.com
518-869-8116

Reveal Analytics, LLC
765 Riverside Drive
Suite 6D
New York, NY 10032
Oumar Nabe, Ph.D., MBA
917-545-4259
onabe@revealanalytics.com

Ridgefield One LLC
79 Danbury Rd, Suite 3A, Ridgefield, CT 06877
David Marceau
203-438-8800 x101
dmarceau@ridgefieldone.com

Scott Technical Services
PO Box 1475 Baldwin NY 11510
Paul Scott - Consultant
pscott@scott-technical.com
Web: www.scott-technical.com
(347)292-8544

SyigmaTechnology Solutions, Inc.
300 West 135th Street, Suite 5J New York, NY 10030
Stuart Holland
917-507-1500
stuart.holland@sygmatechnology.com

TechLink Systems, Inc.
51 East 42nd St - Suite 1600, New York, NY 10017-5451
Mark Balawejder
215-817-2118
mbalawejder@techlinksystems.com

Unique Comp Inc
2708 42nd Road Long Island City, NY 11101
Gautam Tooley
718-392-5100 / 202-297-4889
gtooley@uciny.com

Vichara Technologies, Inc.
5 Marine View Plaza, Suite 312
Hoboken, NJ 07030
Dennis L. Warner
917-608-7218
dwarner@vichara.com

All other terms and conditions remain unchanged.

If submitting a proposal, this Addendum No. 1 for RFQ #1768 must contain an original signature, be dated, attached to, and made a part of your bid. If you already submitted a proposal, you still need to complete and submit this addendum. If you wish to modify your response to an already submitted proposal, you may do so however any such revision must be clearly identified as a revised proposal superseding the previously submitted proposal. All such modifications must be received prior to the bid due date.

Please sign below and submit Addendum No. 1 with your bid package.

Company Name

Address (include City, State, Zip)

Bidders Name (please print)

Title

Signature

Date

ASP.NET Coding Standards and Best Practices

1.0 Introduction.....	1
1.1 Scope.....	1
1.2 Document Conventions.....	1
Coloring & Emphasis:.....	1
Keywords:.....	1
1.3 Terminology & Definitions.....	2
Access Modifier.....	2
Class.....	2
Common Type System.....	2
Function.....	2
Identifier.....	2
Magic Number.....	2
Member Variables.....	2
Module.....	3
Premature Generalization.....	3
Procedure.....	3
Namespace.....	3
Subroutine.....	3
2.0 Naming Conventions and Standards.....	4
2.1 General Guidelines.....	4
2.2 Capitalization Styles.....	4
2.2.1 Pascal case.....	4
2.2.2 Camel case.....	4
2.2.3 Uppercase.....	5
2.3 Acronyms and Abbreviations.....	5
2.4 Case Sensitivity.....	6
2.5 Word Choice.....	7
2.6 Avoiding Type Name Confusion.....	8
2.7 Application, Project, Solution, and Website Names.....	10
2.8 Folder Names.....	10
2.9 File Names.....	10
2.10 Namespaces.....	10
2.11 Classes.....	11
2.12 Procedures (Methods, Functions and Sub-Routines).....	12
2.13 Properties.....	12
2.14 Events & Event Handlers.....	14
2.14.1 Events.....	14
2.14.2 Event Handlers.....	14
2.15 Interfaces.....	15
2.16 Constants.....	15
2.17 Enumerations.....	16
2.18 Static Fields.....	16
2.19 Parameters.....	16
2.20 Member Variables.....	17
2.21 Class Level Private and Protected Variables (Global Variables).....	17
2.22 Class Level Public Variables (Global Variables).....	18
2.23 Procedure Level Local Variables.....	18
2.24 Menus.....	19
2.25 Exceptions.....	19

2.26 Structures	19
2.27 Delegates	20
2.28 Collections	20
2.29 Attributes.....	20
2.30 Standard Control Prefixes	20
2.30.1 Control Objects	20
2.30.2 Database Objects.....	21
2.31 ADO.NET Naming Conventions	22
2.31 Database Naming Conventions.....	22
3.0 Comments and Embedded Documentation.....	23
3.1 General Rules.....	23
3.2 Class Comments.....	23
3.3 Procedure/Method Comments	23
3.4 Special Comments (for development)	23
3.5 Header Blocks.....	24
3.6 Commenting out code vs. removing code.....	24
Detect and remove dead code	24
Detect and remove code that has no effect	24
3.8 XML Tags.....	25
4.0 Coding Style and Practices	25
4.1 General Rules.....	25
4.2 Indentation and Spacing.....	26
4.2.1 Indentation	26
4.2.2 Spacing.....	26
4.3 Procedure Conventions	27
4.4 Region Tags	27
4.5 Menus.....	28
4.5.1 General Rules.....	28
4.5.2 Available and Unavailable Menus and Menu Items	28
4.5.3 Common Menus.....	29
4.9 Designing Classes, Modules and Procedures.....	29
4.9.1 General Rules.....	29
4.9.2 Classes and Modules.....	30
4.9.3 Procedures (Methods, Functions and Sub-Routines).....	30
4.10 Object Life Cycle.....	31
4.10.1 General Rules.....	31
4.10.2 Constant Values	31
4.10.3 Public Static Read-Only Fields	31
4.10.4 Reference Fields.....	32
4.10.5 Shadowing.....	32
4.10.6 Destructors	32
4.10.7 Classes using unmanaged or expensive resources	34
4.10.8 Reference Type Members in the Destructor	35
4.10.9 Returning a Copy of a Reference Type or Array.....	35
4.11 Control Flow	35
4.11.1 Loop Variables.....	35
4.11.2 Flow Control Primitives (if, else, while, for, do, switch)	35
4.11.3 Switch Statements.....	36
4.11.4 Else If Sub-Statement	36

4.11.5 Return Statements.....	36
4.11.6 Comparisons	36
4.11.7 Accessing modified objects	37
4.11.8 Simple Assignment or Initialization	37
4.12 Object Oriented Programming.....	38
4.12.1 General Rules.....	38
4.12.2 Overriding a Method.....	38
4.12.3 Specifying Methods and Classes	39
4.12.4 Describe pre-conditions, post-conditions, exceptions, and class invariants.	39
4.12.5 Referencing an Object of a Derived Class	39
4.12.6 Overloading ‘modifying’ operators on a class type.....	39
4.12.7 Implementation of an Overloaded Operator	40
4.12.8 The operator==(), Equals method and GetHashCode()	40
4.12.9 Structs and Value Semantics.....	40
4.12.10 Properties	40
4.12.11 Methods vs. Properties.....	41
4.12.17 Constructors	41
4.12.18 Operation Results.....	41
4.12.19 Interfaces.....	42
4.13 Various Data Types.....	42
4.13.1 Enumerations	42
4.13.2 The [Flags] attribute on an enumeration.....	43
4.13.3 “Magic Numbers”	43
4.13.4 Floating Point Values.....	44
4.13.5 Casting Types.....	44
4.14 Delegates and Events	44
4.14.1 Object State.....	44
4.14.2 Threads.....	44
4.14.3 Raising Events	45
4.14.4 Sender / Arguments.....	45
4.14.5 Add / Remove Accessors	46
4.14.6 Property-Changed Events	46
4.14.7 Interface vs. Delegate.....	46
4.15 Exception Handling	47
4.15.1 General Rules.....	47
4.15.2 Try/Catch Blocks	48
4.15.3 Throw the most specific exception possible.	49
4.15.4 Throw informational exceptions.....	49
4.15.5 Always log that an exception is thrown.....	49
4.15.6 Only throw exceptions in exceptional situations.	50
4.15.6 Do not throw exceptions from inside destructors.	50
4.15.7 Only re-throw exceptions when you want to specialize the exception.....	50
4.15.8 List the explicit exceptions	51
4.15.9 Allowing callers to prevent exceptions.....	51
4.15.10 Use standard exceptions.....	51
4.15.11 Only catch the exceptions explicitly mentioned in the documentation.	52
4.16 Bracing (C#).....	53
5.0 Data-Specific Programming Guidelines	53
5.1 Data Access.....	53

5.2 User Stored Procedures over inline SQL	54
5.2.1 Rewriting inline SQL statements as Stored Procedures	54
5.2.2 Why use a Stored Procedure rather than in-line SQL.....	54
5.3 Transactions	56
6.0 SQL Reporting Services	57
7.0 Project Settings, Structure, and Architecture.....	58
7.1 Connection Strings.....	59
7.1.1 General Rules.....	59
7.1.2 Web.Config.....	59
8.0 Security	59
8.1 Input/Data Validation.....	59
8.1.1 Do Not Rely on ASP.NET Request Validation	59
8.1.2 Validate Input for Length, Range, Format, and Type.....	60
8.1.3 Validate Input from All Sources Like QueryString, Cookies, and HTML Controls.....	60
8.1.4 Do Not Rely on Client-Side Validation	60
8.1.5 Avoid User-Supplied File Name and Path Input	60
8.1.6 Do Not Echo Untrusted Input	61
8.1.7 If You Need to Write Out Untrusted Data, Encode the Output	61
8.1.8 Additional Resources	61
8.2 Authentication.....	62
8.2.1 Forms Authentication.....	62
8.2.2 Windows Authentication	65
8.3 Authorization	66
8.3.1 Use URL Authorization for Page and Directory Access Control	66
8.3.2 Configure ACLs on Your Web Site Files	67
8.3.3 Use ASP.NET Role Manager for Roles Authorization	67
8.3.4 If Your Role Lookup Is Expensive, Consider Role Caching.....	68
8.3.5 Protect Your Authorization Cookie	68
8.3.6 Additional Resources	69
8.4 Code Access Security	69
8.4.1 Consider Code Access Security for Partial Trust Applications	69
8.4.2 Choose a Trust Level That Does Not Exceed Your Application's Requirements	70
8.4.3 Create a Custom Trust Policy if Your Application Needs Additional Permissions	70
8.4.4 Use Medium Trust in Shared Hosting Environments	70
8.5 Data Access.....	71
8.5.1 Encrypt your connection strings	71
8.5.2 Use Least-Privileged Accounts for Database Access	71
8.5.3 Use Windows Authentication Where Possible	72
8.5.4 If You Use Windows Authentication, Use a Trusted Service Account.....	72
8.5.5 If You Cannot Use a Domain Account, Consider Mirrored Accounts	72
8.5.6 When Using SQL Authentication, Use Strong Passwords	72
8.5.7 When Using SQL Authentication, Protect Credentials Over the Network.....	73
8.5.8 When Using SQL Authentication, Protect Credentials in Configuration Files	73
8.5.9 Validate Untrusted Input Passed to Your Data Access Methods.....	73
8.5.10 When Constructing SQL Queries, Use Type Safe SQL Parameters	74
8.5.11 Avoid Dynamic Queries That Accept User Input.....	74
8.5.12 Additional Resources	74
8.6 Exception Management	75
8.6.1 Use Structured Exception Handling.....	75

8.6.2 Do Not Reveal Exception Details to the Client	75
8.6.3 Use a Global Error Handler to Catch Unhandled Exceptions.....	76
8.7 Impersonation/Delegation.....	76
8.7.1 Know Your Tradeoffs with Impersonation.....	76
8.7.2 Avoid Calling LogonUser.....	77
8.7.3 Avoid Programmatic Impersonation Where Possible.....	77
8.7.4 If You Need to Impersonate, Consider Threading Issues	77
8.7.5 If You Need to Impersonate, Clean Up Appropriately	78
8.7.6 Avoid Losing Impersonation Tokens.....	78
8.8 Parameter Manipulation.....	79
8.8.1 Do Not Make Security Decisions Based on Parameters Accessible on the Client-Side	79
8.8.2 Validate All Input Parameters.....	79
8.8.3 Avoid Storing Sensitive Data in ViewState.....	79
8.8.4 Encrypt ViewState if It Must Contain Sensitive Data	80
8.8.5 Additional Considerations	80
8.9 Sensitive Data	80
8.9.1 Avoid Plaintext Passwords in Configuration Files	81
8.9.2 Use Platform Features to Manage Keys Where Possible.....	81
8.9.3 Do Not Pass Sensitive Data from Page to Page.....	81
8.9.4 Protect Sensitive Data Over the Wire	81
8.9.5 Do Not Cache Sensitive Data	81
8.10 Session Management	81
8.10.1 Do Not Rely on Client-Side State Management Options	81
8.10.2 Protect Your Out-of-Process State Service.....	81
8.10.3 Protect SQL Server Session State	82
8.11 Auditing and Logging.....	82
8.11.1 Use Health Monitoring to Log and Audit Events	83
8.11.2 Instrument for User Management Events	83
8.11.3 Instrument for Unusual Activity	83
8.11.4 Instrument for Significant Business Operations	83
8.11.5 Consider Using an Application-specific Event Source.....	84
8.11.6 Protect Audit and Log Files	84
8.11.7 Additional Considerations	84
8.12 Deployment Considerations.....	84
8.12.1 Use a Least-Privileged Account for Running ASP.NET Applications.....	85
8.12.2 Encrypt Configuration Sections That Store Sensitive Data.....	85
8.12.3 Consider Your Key Storage Location.....	86
8.12.4 Block Protected File Retrieval by Using HttpForbiddenHandler	86
8.12.5 Configure the MachineKey to Use the Same Keys on All Servers in a Web Farm	86
8.12.6 Lock Configuration Settings to Enforce Policy Settings	87
8.13 Communication Security	87
8.13.1 Consider SSL vs. IPsec	87
8.13.2 Optimize Pages That Use SSL.....	88
9.0 Appendix A.....	89
Include files in .Net.....	89
When to use ASP.NET includes	89
Using an ASP.NET include	89
Naming your ASP.NET include files.....	90
How To Dynamically Include Files in ASP.NET.....	90

MORE INFORMATION	91
10.0 Appendix B	92
ASP.NET 2.0 Web.Config.....	92
Case-Sensitivity	92
Custom Sections.....	92
Configuring Specific Files and Subdirectories	92
Database Connections:.....	94
Session States:.....	94
Error Handling:	95
Security:	95
11.0 References.....	97

1.0 Introduction

This document describes rules and recommendations for developing applications and class libraries using VB.NET and C# Languages. The objective of this document is to define the coding standards to be utilized when programming applications for the New York State Office of General Services in order to enforce consistent style and formatting. The positive effects are:

- Avoidance of errors/bugs,
- Ease of maintainability, by promoting some proven design principles,
- Ease of maintainability, by requiring or recommending a certain unity of style,
- Ease of extensibility,
- Performance, by dissuading wasteful practices,
- Enhanced Readability and inherent documentation,
- Consistency of code.

Specifically, this document covers Naming Conventions, Coding Style, Language Usage, and Object Model Design. These standards do not list every possible object or control.

1.1 Scope

This document applies to the VB.NET Language, the C# Language and the .NET Framework Common Type System (CTS) it implements.

This document does not address usage of .NET Framework class libraries.

The standards for curly-braces ({ or }) and white space (tabs vs. spaces) are addressed here to ensure greater consistency and maintainability of source code.

1.2 Document Conventions

Much like the ensuing coding standards, this document requires standards in order to ensure clarity when stating the rules and guidelines. Certain conventions are used throughout this document to add emphasis.

Below are some of the common conventions used throughout this document.

Coloring & Emphasis:

Blue Text: colored blue indicates a C# keyword.

Green Text: colored green indicates a VB.NET keyword.

Bold Text: with additional emphasis to make it stand-out.

Keywords:

Always: Emphasizes this rule must be enforced.

Never: Emphasizes this action must not happen.

Do Not: Emphasizes this action must not happen.

Avoid: Emphasizes that the action should be prevented, but some exceptions may exist.

Try: Emphasizes that the rule should be attempted whenever possible and appropriate.

Use: Emphasizes that the rule should be attempted whenever possible and appropriate.

Consider: Emphasizes that the rule should be attempted whenever possible and appropriate.

Example: Precedes text used to illustrate a rule or recommendation.

Reason: Explains the thoughts and purpose behind a rule or recommendation.

1.3 Terminology & Definitions

The following terminology is referenced throughout this document:

Access Modifier

The C# keywords **public**: (No access restrictions.), **protected**: (Access limited to containing/derived classes.), **internal**: (Access limited to containing type.), **protected internal**: (Access limited to the current project.), **private**: (Access limited to containing/derived classes and the current project.) and the VB.NET keywords **Public**: (No access restrictions.), **Protected**: (Access limited to containing/derived classes.), **Private**: (Access limited to containing type.), **Friend**: (Access limited to the current project.), **Protected Friend**: (Access limited to containing/derived classes and the current project.) declare the allowed code-accessibility of types and their members. Although default access modifiers vary, classes and most other members use the default of private. Notable exceptions are *interfaces* and *enums* which both default to public.

Class

Templates used for defining new types. Classes describe both the *properties* and *behaviors* of objects. *Properties* contain the data that are exposed by the class. *Behaviors* are the functionality of the object, and are defined by the public methods (also called member functions) and events of the class. Collectively, the public properties and methods of a class are known as the *object interface*. Classes themselves are not objects, but instead they are used to instantiate (i.e., create) objects in memory.

Common Type System

The .NET Framework common type system (CTS) defines how types are declared, used, and managed. All native C# types are based upon the CTS to ensure support for cross-language integration.

Function

A named section of a program that performs a specific task. In this sense, a function is a type of procedure or routine. Some programming languages make a distinction between a *function*, which returns a value, and a *procedure*, which performs some operation but does not return a value.

Identifier

"Identifiers" or "symbols" are developer defined tokens used to uniquely name a declared object or object instance such as *variables*, *types*, *functions*, *subroutines*, *packages*, and *labels* which name language entities in a program. Identifier names must differ in spelling and case from any keywords. Keywords cannot be used as identifiers; they are reserved for special use. You create an identifier by specifying it in the declaration of a variable, type, or function.

In C#, identifiers are also used to give names to the 'tags' of *structures*, *unions*, and *enums*.

Visual Basic .NET identifiers conform to the Unicode Standard Annex 15 with one exception: identifiers may begin with an underscore (`_`) character. If an identifier begins with an (`_`), it must contain at least one other valid identifier character to disambiguate it from a line continuation.

Example: `result` is an identifier for an integer variable and `main` and `printf` are identifier names for *functions*.

Example: `public class MyClasMyClassNameIdentifier { ... }`

Magic Number

Any numeric literal used within an expression (or to initialize a variable) that does not have an obvious or well known meaning. This usually excludes the integers 0 or 1 and any other numeric equivalent precision that evaluates as zero.

Member Variables

Variable part of a class or object, i.e. class or instance variables. Does not include local variables.

Module

A module is a software entity that groups a set of (typically cohesive) subprograms/routines and data structures. Modules are units that can be compiled separately, which makes them reusable and allows multiple programmers to work on different modules simultaneously. In this respect they are similar to objects in an object-oriented language, though a module may contain many procedures and/or functions which would correspond too many objects. Modules also promote modularity and encapsulation (i.e. information hiding), both of which can make complex programs easier to understand. Programs are composed of one or more independently developed modules that are not combined until the program is linked. A module often has its own name space for identifiers so the same identifier may be used to mean different things in different modules.

Premature Generalization

As it applies to object model design; this is the act of creating abstractions within an object model not based upon concrete requirements or a known future need for the abstraction. In simplest terms: "Abstraction for the sake of Abstraction."

Procedure

A sequence of instructions for performing a particular task. It is also known as a *routine*, *subroutine*, or *function*. Most programming languages, including most machine languages, allow the programmer to define subroutines. This allows the subroutine code to be called from multiple places, even from within itself (in which case it is called recursive). The programming language implementation takes care of returning control to (just after) the calling location, usually with the support of call and return instructions at machine language level.

Most languages also allow arguments to be passed to the subroutine, and one, or occasionally more, return values to be passed back.

A function is often very similar to a subroutine, the main difference being that it is called chiefly for its return value, rather than for any side effects.

Namespace

An abstract container or environment created to hold a logical grouping of unique identifiers (i.e., names). An identifier defined in a namespace is associated with that namespace. The same identifier can be independently defined in multiple namespaces. That is, the meaning associated with an identifier defined in one namespace may or may not have the same meaning as the same identifier defined in another namespace.

Subroutine

A sequence of computer instructions for performing a specified task that can be used repeatedly. A large subroutine might be called a "*module*" or "*procedure*." The subroutine code can be called from multiple places, even from within itself (in which case it is called recursive). The programming language implementation takes care of returning control to (just after) the calling location, usually with the support of call and return instructions at machine language level.

2.0 Naming Conventions and Standards

2.1 General Guidelines

Follow all .NET Framework Design Guidelines for both internal and external members. Highlights of these include:

- **Use** US-English for naming identifiers.
- **Avoid** using abbreviations. If they must be used, use only well known abbreviations.
- **Do not** use letters that can be mistaken for digits, and vice versa.
Example: Capital O and the number 0 or lowercase l and the number 1.
- **Use** [Pascal](#) and [Camel](#) casing for naming identifiers.
- **Do not** use Hungarian notation or add any other type identification to identifiers.
- The reasons to extend the public rules (no Hungarian, no prefix for member variables, etc.) are to produce a consistent source code appearance. In addition a goal is to have clean readable source. Code legibility should be a primary goal.
- **Do not** prefix member fields.
- **Do not** use casing to differentiate identifiers.
- **Do not** use the underscore character (`_`) in identifiers.
- Name an identifier according to its meaning and not its type.
- **Do not** add code-archive related prefixes to identifiers.
Example: `oldPage_Load`
- Only use the **this**. construction in C# and **Me**. construction in VB to avoid a name clash.
- **Do not** use "My" or "my" as part of a variable name. This creates confusion with the **My** objects.

2.2 Capitalization Styles

Capitalize every word except articles (“a,” “an,” and “the”), coordinating conjunctions (i.e., “and,” “or,” “but,” “so,” “yet,” and “nor”), and prepositions with fewer than four letters (like “in”).

Use the following three conventions for capitalizing identifiers:

2.2.1 Pascal case

- The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. You can use **Pascal Case** for identifiers of three or more characters.
Example: `BackColor` or `CustomerName`
- **Use Pascal Case** on all Methods, Functions, Sub-Routines, Events, Enumeration types and values, Static fields, Interfaces, Namespaces, Properties and Class names

2.2.2 Camel case

- The first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalized.
Example: `backColor` or `customerName`
- **Use Camel Case** on all variable declarations.

2.2.3 Uppercase

- **All Constants** must be in all uppercase.
- Capitalize all letters in the identifier only for identifiers that consist of two or fewer letters.

Example: System.**IO** or System.Web.**UI**

- You might also have to capitalize identifiers to maintain compatibility with existing, unmanaged symbol schemes, where all uppercase characters are often used for enumerations and constant values. In general, these symbols should not be visible outside of the assembly that uses them.

The following table summarizes the capitalization rules and provides examples for the different types of identifiers.

Identifier	Case	Example
Namespace	Pascal	System.Drawing
Class	Pascal	AppDomain
Method	Pascal	ToString
Property	Pascal	BackColor
Event	Pascal	ValueChanged
Parameter	Camel	typeName
Exception class	Pascal	WebException
Note: Always ends with the suffix Exception.		
Read-only Static field	Pascal	RedValue
Enum type	Pascal	ErrorLevel
Enum values	Pascal	FatalError
Interface	Pascal	IDisposable
Note: Always begins with the prefix I.		
Protected instance field	Camel	redValue
Note: Rarely used. A property is preferable to using a protected instance field.		
Public instance field	Pascal	RedValue
Note: Rarely used. A property is preferable to using a public instance field.		

2.3 Acronyms and Abbreviations

To avoid confusion and guarantee cross-language interoperability, follow these rules regarding the use of abbreviations:

- **Do not** use abbreviations or contractions as parts of identifier or parameter names.
Example: Use **GetWindow** instead of **GetWin**.
- **Do not** use acronyms that are not generally accepted in the computing field.
- If you must use abbreviations, use **Camel Case** for abbreviations that consist of more than two characters, even if this contradicts the standard abbreviation of the word. Two character abbreviations at the start of an identifier are written in lower case for **Camel casing**.

Examples:

CAMEL CASING	PASCAL CASING
newItem	NewItem
uiForm	UIForm
ogsIT	OgsIT

- **Use** well-known acronyms to replace lengthy phrase names.
Example: Use **UI** for User Interface and **OLAP** for On-line Analytical Processing.
- **Use** **Pascal Case** or **Camel Case** for acronyms more than two characters long.
Example: Use **HtmlButton** or **htmlButton**. However, you must capitalize acronyms that consist of only two characters, such as **System.IO** instead of **System.io**.

2.4 Case Sensitivity

To avoid confusion and guarantee cross-language interoperability, follow these rules regarding the use of case sensitivity:

- **Do not** use names that require case sensitivity. Components must be fully usable from both case-sensitive and case-insensitive languages. Case-insensitive languages like VB.NET cannot distinguish between two names within the same context that differ only by case. Therefore, you must avoid this situation in the components or classes that you create.

- **Do not** create two namespaces with names that differ only by case.

A case insensitive language like VB.NET cannot distinguish between the following two namespace declarations.

[C#]

Example: namespace **ee.cummings;**

Example: namespace **Ee.Cummings;**

- **Do not** create a function with parameter names that differ only by case.

The following example is incorrect.

[C#]

Example: void MyFunction(string **a**, string **A**)

- **Do not** create a namespace with type names that differ only by case.

In the following example, **Point p** and **POINT p** are inappropriate type names because they differ only by case.

[C#]

Example: System.Windows.Forms.Point **p**

Example: System.Windows.Forms.POINT **p**

- **Do not** create a type with property names that differ only by case.

In the following example, **int Color** and **int COLOR** are inappropriate property names because they differ only by case.

[C#]

Example: int Color {get, set}

Example: int COLOR {get, set}

- **Do not** create a type with method names that differ only by case.

In the following example, **calculate** and **Calculate** are inappropriate method names because they differ only by case.

[C#]

Example: void calculate()

Example: void Calculate()

2.5 Word Choice

- **Avoid** using Class names that duplicate commonly used .NET Framework namespaces.

Example: Do not use any of the following names as a Class name: **System, Collections, Forms, or UI**. See the Class Library for a list of .NET Framework namespaces.

- **Avoid** using identifiers that conflict with the following keywords.

AddHandler	AddressOf	Alias	And	Ansi
As	Assembly	Auto	Base	Boolean
ByRef	Byte	ByVal	Call	Case
Catch	CBool	CByte	CChar	CDate
CDec	CDBl	Char	CInt	Class
CLng	CObj	Const	CShort	CSng
CStr	CType	Date	Decimal	Declare
Default	Delegate	Dim	Do	Double
Each	Else	Elseif	End	Enum
Erase	Error	Event	Exit	ExternalSource
False	Finalize	Finally	Float	For
Friend	Function	Get	GetType	Goto
Handles	If	Implements	Imports	In
Inherits	Integer	Interface	Is	Let
Lib	Like	Long	Loop	Me
Mod	Module	MustInherit	MustOverride	MyBase
MyClass	Namespace	New	Next	Not
Nothing	NotInheritable	NotOverridable	Object	On
Option	Optional	Or	Overloads	Overridable
Overrides	ParamArray	Preserve	Private	Property
Protected	Public	RaiseEvent	ReadOnly	ReDim
Region	REM	RemoveHandler	Resume	Return
Select	Set	Shadows	Shared	Short
Single	Static	Step	Stop	String
Structure	Sub	SyncLock	Then	Throw
To	True	Try	TypeOf	Unicode
Until	volatile	When	While	With
WithEvents	WriteOnly	Xor	eval	extends
instanceof	package	var		

2.6 Avoiding Type Name Confusion

Different programming languages use different terms to identify the fundamental managed types. Class library designers must avoid using language-specific terminology. Follow the rules described in this section to avoid type name confusion.

- **Use** names that describe a type's **meaning** rather than names that describe the **type**. In the rare case that a parameter has no semantic meaning beyond its type, use a generic name.

Example: A class that supports writing a variety of data types into a stream might have the following methods.

[VB.NET]

Sub Write(value As Double)	Sub Write(value As Single)	Sub Write(value As Long)
Sub Write(value As Integer)	Sub Write(value As Short)	

[C#]

void Write(double value);	void Write(float value);	void Write(long value);
void Write(int value);	void Write(short value);	

- **Do not** create language-specific method names, as in the following example.

[VB.NET]

Sub Write(doubleValue As Double)	Sub Write(singleValue As Single)	Sub Write(longValue As Long)
Sub Write(integerValue As Integer)	Sub Write(shortValue As Short)	

[C#]

void Write(double doubleValue);	void Write(float floatValue);	void Write(long longValue);
void Write(int intValue);	void Write(short shortValue);	

- In the extremely rare case that it is necessary to create a uniquely named method for each fundamental data type, use a universal type name. The following table lists fundamental data type names and their universal substitutions.

C# type name	Visual Basic type	JScript type name	Universal type name
sbyte	SByte	sByte	SByte
byte	Byte	byte	Byte
short	Short	short	Int16
ushort	UInt16	ushort	UInt16
int	Integer	int	Int32
uint	UInt32	uint	UInt32
long	Long	long	Int64
ulong	UInt64	ulong	UInt64
float	Single	float	Single
double	Double	double	Double
bool	Boolean	boolean	Boolean
char	Char	char	Char
string	String	string	String
object	Object	object	Object

Example: A class that supports reading a variety of data types from a stream might have the following methods.

[VB.NET]

ReadDouble() <i>As Double</i>	ReadSingle() <i>As Single</i>	ReadInt64() <i>As Long</i>
ReadInt32() <i>As Integer</i>	ReadInt16() <i>As Short</i>	

[C#]

double ReadDouble();	float ReadSingle();	long ReadInt64();
int ReadInt32();	short ReadInt16();	

The preceding example is preferable to the following language-specific alternative.

[VB.NET]

ReadDouble() <i>As Double</i>	ReadSingle() <i>As Single</i>	ReadLong() <i>As Long</i>
ReadInteger() <i>As Integer</i>	ReadShort() <i>As Short</i>	

[C#]

double ReadDouble();	float ReadFloat();	long ReadLong();
int ReadInt();	short ReadShort();	

2.7 Application, Project, Solution, and Website Names

- Use [Pascal Case](#) for naming Application files, Project files, Solution files, and Website files.
- Adhere to the Naming Convention General Guidelines. **See Section 2.1.**

2.8 Folder Names

- Use [Pascal Case](#) for naming folders.
- Adhere to the Naming Convention General Guidelines. **See Section 2.1.**

2.9 File Names

- Use [Pascal Case](#) for naming source code files.
- Name the source code file to the class name.
Example: A file name of Default.vb and a class name of Default.
- Adhere to the Naming Convention General Guidelines. **See Section 2.1.**

2.10 Namespaces

The general rule for naming namespaces is to use according to a well-defined pattern utilizing the company name followed by the technology name and optionally the feature and design as follows.

- **CompanyName.TechnologyName[.Feature][.Design]**
Example: Microsoft.Media
Example: Microsoft.Media.Design
- Prefixing namespace names with a company name or other well-established brand avoids the possibility of two published namespaces having the same name.
Example: **Microsoft.Office** is an appropriate prefix for the Office Automation Classes provided by Microsoft.
- **Use** a stable, recognized technology name at the second level of a hierarchical name. Use organizational hierarchies as the basis for namespace hierarchies. Name a namespace that contains types that provide design-time functionality for a base namespace with the .Design suffix.
Example: The **System.Windows.Forms.Design** Namespace contains designers and related classes used to design **System.Windows.Forms** based applications.
- A nested namespace must have a dependency on types in the containing namespace.
Example: The classes in the **System.Web.UI.Design** depend on the classes in **System.Web.UI**. However, the classes in **System.Web.UI** do not depend on the classes in **System.Web.UI.Design**.
- **Use Pascal Case** for namespaces, and separate logical components with periods, as in **Microsoft.Office.PowerPoint**. If your brand employs nontraditional casing, follow the casing defined by your brand, even if it deviates from the prescribed [Pascal Case](#).
Example: The namespaces **NeXT.WebObjects** and **ee.cummings** illustrate appropriate deviations from the Pascal case rule.
- **Use** plural namespace names if it is semantically appropriate.
Example: Use **System.Collections** rather than **System.Collection**.
- Exceptions to using plural namespace names are brand names and abbreviations.
Example: Use **System.IO** rather than **System.IOs**.
- **Do not** use the same name for a namespace and a class.
Example: Do not provide both a **Debug** namespace and a **Debug** class.
- **Always** name DLL assemblies after their containing namespace.
Example: If you name an assembly **MyCompany.MyTechnology.dll** then the namespace must be **MyCompany.MyTechnology**.

2.11 Classes

The following rules outline the guidelines for naming classes:

- Use Pascal Case for class names
- Use a noun or noun phrase to name a class.
- Do not use the underscore character (_) for class names.
- Avoid classes having the same name as the namespace in which they reside.
- Avoid using abbreviations. If they must be used, use only well known abbreviations.
- Do not prefix classes with any letter. Do not use a type prefix, such as “C” or “cls” for class, on a class name.
Example: Use the class name **FileStream** rather than **CFileStream**.
- Do not add a suffix like “Class” to a class name (Exceptions: Attributes and Exceptions).
- Occasionally, it is necessary to provide a class name that begins with the letter I, even though the class is not an interface. This is appropriate as long as I is the first letter of an entire word that is a part of the class name.
Example: The class name **IdentityStore** is appropriate.
- Consider using a compound word to name a derived class. The second part of the derived class's name must be the name of the base class.
Example: **ApplicationException** is an appropriate name for a class derived from a class named **Exception**, because **ApplicationException** is a kind of **Exception**.
- Use reasonable judgment when using a compound word to name a derived class.
Example: **Button** is an appropriate name for a class derived from **Control**. Although a button is a kind of control, making Control a part of the class name would lengthen the name unnecessarily.

The following are examples of correctly named classes.

[VB.NET]

Public Class FileStream	Public Class Button	Public Class String
-------------------------	---------------------	---------------------

[C#]

public class FileStream	public class Button	public class String
-------------------------	---------------------	---------------------

2.12 Procedures (Methods, Functions and Sub-Routines)

The following rules outline the naming guidelines for procedures:

- Use **Pascal Case** for procedure names.
- Use verbs or verb phrases to name procedure that are descriptive of what action a procedure performs.
- **Always** be consistent in naming procedures.
- **Do not** use the underscore character (_) for procedure names except in the event handlers.
- **Avoid** using abbreviations. If they must be used, use only well known abbreviations.
- Functions and subs must differ by more than case to be usable from case-insensitive languages.

[VB.NET]

Example: `Public Sub DoSomething(...)`

[C#]

Example: `public void DoSomething(...)`

- **Do not** prefix methods with any letter.
- **Do not** add a Callback or similar suffix to callback methods.

The following are examples of correctly named methods.

`RemoveAll()`

`GetCharArray()`

`Invoke()`

2.13 Properties

The following rules outline the naming guidelines for properties:

- Use **Pascal Case** for property names.
- Use a noun or noun phrase to name properties.
- **Do not** use the underscore character (_) for property names.
- **Avoid** using abbreviations. If they must be used, use only well known abbreviations.
- **Do not** use Hungarian notation.
- **Consider** creating a property with the same name as its underlying type.
Example: If you declare a property named **Color**, the type of the property must likewise be **Color**. See the example later in this topic.

The following code example illustrates correct property naming.

[VB.NET]

```
Public Class SampleClass
    Public Property BackColor As Color
        ' Code for Get and Set accessors goes here.
    End Property
End Class
```

[C#]

```
public class SampleClass
{
    public Color BackColor
    {
        // Code for Get and Set accessors goes here.
    }
}
```

The following code example illustrates providing a property with the same name as a type.

```
[VB.NET]  
Public Enum Color  
    ' Insert code for Enum here.  
End Enum  
Public Class Control  
    Public Property Color As Color  
        Get  
            ' Insert code here.  
        End Get  
        Set  
            ' Insert code here.  
        End Set  
    End Property  
End Class
```

```
[C#]  
public enum Color  
{  
    // Insert code for Enum here.  
}  
public class Control  
{  
    public Color Color  
    {  
        get { // Insert code here. }  
        set { // Insert code here. }  
    }  
}
```

The following code example is **incorrect** because the property Color is of type Integer.

```
[VB.NET]  
Public Enum Color  
    ' Insert code for Enum here.  
End Enum  
Public Class Control  
    Public Property Color As Integer  
        Get  
            ' Insert code here.  
        End Get  
        Set  
            ' Insert code here.  
        End Set  
    End Property  
End Class
```

```
[C#]  
public enum Color { // Insert code for Enum here. }  
public class Control  
{  
    public int Color  
    {  
        get { // Insert code here. }  
        set { // Insert code here. }  
    }  
}
```

In the incorrect example, it is not possible to refer to the members of the Color enumeration. **Color.Xxx** will be interpreted as accessing a member that first gets the value of the Color property (type **Integer** in Visual Basic or type **int** in C#) and then accesses a member of that value (which would have to be an instance member of **System.Int32**).

2.14 Events & Event Handlers

2.14.1 Events

The following rules outline the naming guidelines for events:

- Use **Pascal Case** for event names
- Use an action verb (gerund) (the "ing" form of a verb) to describe an event name that expresses the concept of pre-event, and a past-tense verb ("ed") to represent post-event.
Example: Correctly named event names include **Clicked**, **Painting**, and **DroppedDown**.
Example: A **Close** event that can be canceled must have a **Closing** event and a **Closed** event. Do not use the **BeforeXxx/AfterXxx** naming pattern.
- Do not use Hungarian notation.
- Do not use a prefix or suffix on the event declaration on the type.
Example: Use **Close** instead of **OnClose**.
- Do not add an "Event" suffix (or any other type-related suffix) to the name of an event.
- Always name an event argument class with the **EventArgs** suffix.

2.14.2 Event Handlers

- Always prefix an event handler with **On**.
- Always specify two parameters named **sender** and **e**. The **sender** parameter represents the object that raised the event.

The **sender** parameter is always of type object, even if it is possible to use a more specific type. The state associated with the event is encapsulated in an instance of an event class named **e**. Use an appropriate and specific event class for the **e** parameter type.
- Consider providing a protected method called **OnXxx** on types with events that can be overridden in a derived class. This method should only have the event parameter **e**, because the **sender** is always the instance of the type.

The following example illustrates an event handler with an appropriate name and parameters.

[VB.NET]

```
Public Delegate Sub MouseEventHandler(sender As Object, e As MouseEventArgs)
```

[C#]

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

2.15 Interfaces

The following rules outline the naming guidelines for interfaces:

- Use [Pascal Case](#) for Interface names.
- **Always** prefix interface names with the letter I, to indicate that the type is an interface.
- **Always** name interfaces with nouns or noun phrases, or adjectives that describe behavior.
Example: The interface name **IComponent** uses a descriptive noun. The interface name **ICustomAttributeProvider** uses a noun phrase. The name **IPersistable** uses an adjective.
- **Do not** use the underscore character (_) for Interface names.
- **Avoid** using abbreviations. If they must be used, use only well known abbreviations.
- **Use** similar names for the default implementation of an interface.
- **Use** similar names when you define a class/interface pair where the class is a standard implementation of the interface. The names must differ only by the letter I prefix on the interface name.

The following are examples of correctly named interfaces.

[VB.NET]

```
Public Interface IServiceProvider
Public Interface IFormatable
```

[C#]

```
public interface IServiceProvider
public interface IFormatable
```

The following code example illustrates how to define the interface IComponent and its standard implementation, the class Component.

[VB.NET]

```
Public Interface IComponent
    ' Implementation code goes here.
End Interface
Public Class Component
    Implements IComponent
    ' Implementation code goes here.
End Class
```

[C#]

```
public interface IComponent
{
    ' Implementation code goes here.
}
public class Component: IComponent
{
    ' Implementation code goes here.
}
```

2.16 Constants

Constants help to make code more readable.

- Use [Pascal Case](#) if a constant is exposed **publicly** from a class.
Example: SomeClass.SomePublicConstant
- Use all [Uppercase](#) with an underscore (_) to separate words if a constant is **private** to a procedure.
Example: Private Const DATABASE_PATH As String

2.17 Enumerations

The enumeration (Enum) value type inherits from the Enum Class. The following rules outline the naming guidelines for enumerations:

- Use [Pascal Case](#) for Enum types and value names.
- Do not prefix enumerations with any letter.
- Avoid using abbreviations. If they must be used, use only well known abbreviations.
- Do not add an Enum suffix to enumeration type names.
- Use the **FlagsAttribute** custom attribute for an enumeration only if a bitwise operation (AND, OR, EXCLUSIVE OR) is to be performed on a numeric value.
- Use singular names for most enumeration types.
- Use a plural name for enumerations representing bit fields.

2.18 Static Fields

The following rules outline the naming guidelines for static fields:

- Use [Pascal Case](#) for static variable names.
- Use nouns, noun phrases, or abbreviations of nouns to name static fields.
- Do not use a Hungarian notation prefix on static field names.
- Use static properties instead of public static fields whenever possible.

2.19 Parameters

It is important to carefully follow these parameter naming guidelines because visual design tools that provide context sensitive help and class browsing functionality display method parameter names to users in the designer.

The following rules outline the naming guidelines for parameters:

- Use [Camel Case](#) for parameter names.
- Do not prefix parameter names with Hungarian type notation.
- Use descriptive parameter names. Parameter names must be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios.
Example: Visual design tools that provide context sensitive help display method parameters to the developer as they type. The parameter names must be descriptive enough in this scenario to allow the developer to supply the correct parameters.
- Use names that describe a parameter's meaning rather than names that describe a parameter's type. Development tools should provide meaningful information about a parameter's type. Therefore, a parameter's name can be put to better use by describing meaning. Use type-based parameter names sparingly and only where it is appropriate.
- Do not use reserved parameters. Reserved parameters are private parameters that might be exposed in a future version if they are needed. Instead, if more data is needed in a future version of your class library, add a new overload for a method.

The following are examples of correctly named parameters.

[VB.NET]

```
GetType(typeName As String)As Type  
Format(format As String, args() As object)As String
```

[C#]

```
Type GetType(string typeName)  
string Format(string format, object[] args)
```

The following example illustrates a correctly named event argument class.

[VB.NET]

```
Public Class MouseEventArgs
    Inherits EventArgs
    Dim x As Integer
    Dim y As Integer
    Public Sub New MouseEventArgs(x As Integer, y As Integer)
        me.x = x
        me.y = y
    End Sub

    Public Property X As Integer
        Get
            Return x
        End Get
    End Property

    Public Property Y As Integer
        Get
            Return y
        End Get
    End Property
End Class
```

[C#]

```
public class MouseEventArgs : EventArgs
{
    int x;
    int y;
    public MouseEventArgs(int x, int y)
    { this.x = x; this.y = y; }
    public int X { get { return x; } }
    public int Y { get { return y; } }
}
```

2.20 Member Variables

Use [Camel Case](#) for member variable names.

Do not use a prefix for member variables (`_`, `m_`, `s_`, etc.).

Use **"this."** in C# and **"Me."** in VB.NET to distinguish between local and member variables.

2.21 Class Level Private and Protected Variables (Global Variables)

- Use [Camel Case](#) with a leading underscore (`_`) for class level private and protected variable names.
- Always indicate "Protected" or "Private" in VB.NET, **do not** use "Dim".
- The use of "m_" is discouraged, as is use of a variable name that differs from the property by only case, especially with protected variables as that violates compliance.

[VB.NET]

Example: `Private _userID As Integer`

[C#]

Example: `private int _userID ;`

2.22 Class Level Public Variables (Global Variables)

Variables should always be defined with the smallest scope possible. Global (Public) variables can create complex management problems and make the application very difficult to understand. Global variables also make the reuse and maintenance of the code much more difficult. It is highly suggested that programmers **avoid** the use of global variables.

Avoid using Public class-level variables. They are universally frowned upon. It is considered to be a much better practice to **use** property procedures (accessors and mutators) to provide read and/or write access to a private member variable.

If you must expose a member variable to other classes using "Public", follow the property naming conventions:

- **Use Pascal Case** for class level Public variable names.
- **Always** explicitly declare variables.
- **Always** give variables descriptive names.
- **Always** declare variables with carefully chosen datatypes.
- **Do not** use the underscore character (`_`) for variable names.
- **Avoid** using abbreviations. If abbreviations must be used, be consistent in their use. An abbreviation must have only one meaning. If using *min* to abbreviate *minimum*, do so everywhere and do not later use it to abbreviate *minute*. **See Section 2.3** for Abbreviation guidelines.
- **Use** positive form **Is** (which implies Yes/No or True/False values) for Boolean variable names.
Example: PrintJobsFinished.
- **Use** the object data type only when absolutely necessary.
- Members must differ by more than case to be usable from case-insensitive languages.

[VB.NET]

Example: `Public UserID As Integer`

[C#]

Example: `public int UserID`

2.23 Procedure Level Local Variables

Variable names must be short but meaningful. The choice of the variable name must be mnemonic – that is, designed to indicate to the casual observer the intent of its use.

- **Use Camel Case** for local variable names.
- **Always** explicitly declare variables.
- **Always** give variables descriptive names.
- **Always** declare variables with carefully chosen datatypes.
- **Do not** use the underscore character (`_`) for variable names.
- **Avoid** using abbreviations. If abbreviations must be used, be consistent in their use. An abbreviation must have only one meaning. If using *min* to abbreviate *minimum*, do so everywhere and do not later use it to abbreviate *minute*. **See Section 2.3** for Abbreviation guidelines.
- **Use** positive form **Is** (which implies Yes/No or True/False values) for Boolean variable names.
Example: PrintJobsFinished.
- **Use** the object data type only when absolutely necessary.
- **Avoid** the use of one character variable names except in the circumstance of short-lived temporary variables such as **i, j, k, m, and n** for **integers** and **c, d, and e**, for **characters** used in loops and indexing.
- **Avoid** using **temp** as part of a variable name or as a variable name itself, unless it has a very short life span (only a few lines of code).

- **Avoid** using terms such as Flag when naming status variables, which differ from Boolean variables in that they may have more than two possible values.
Example: Instead of documentFlag, use a more descriptive name such as documentFormatType.
- Other common exceptions to the rules for naming local level procedure variables are Exceptions (ex) and Datasets (ds) and Data Adapters (da); they can be used as is without extra naming required within a local scope.

2.24 Menus

Applications frequently use many menu controls, making it useful to have a unique set of naming conventions for these controls.

- **Always** prefix all menu items with “mnu”.
- **Always** extend menu control prefixes beyond the initial "mnu" label by adding an additional prefix for each level of nesting, with the final menu caption at the end of the name string.
When this naming convention is used, all members of a particular menu group are listed next to each other in Visual Basic’s Properties window. In addition, the menu control names clearly document the menu items to which they are attached.

- **Always** name menus in a manner that makes clear the path taken to them.

Example:

If your documentation would describe the path the menu as “File -> Print -> Select Printer”

Then the three menus in question would be named:

```
mnuFile
mnuFilePrint
mnuFilePrintSelect
```

Note: The menu labels do not repeat the name of the parent menu. Opening a menu is like building a sentence.

2.25 Exceptions

- **Always** suffix exception classes with the word “Exception”.

Example:

[VB.NET]

```
Public Class EmployeeListNotFoundException
    Inherits Exception
```

- Follow the naming standards for Classes.

2.26 Structures

- **Use** a noun or a noun phrase to name a structure.
- **Do not** use the underscore character (_) for structure names.
- **Avoid** using abbreviations. If they must be used, use only well known abbreviations.
- **Do not** prefix structure with any letter.
- **Do not** add a suffix like “Struct” to a structure name.

2.27 Delegates

- Follow the naming standards for Classes.
- **Do not** prefix delegates with any letter.
- **Always** add the suffix “Delegate” to the end of the name.
- **Always** add the suffix “EventHandler” to delegates related to events.
- **Always** add the suffix “Callback” to delegates related to callback methods.
Example: MyCallbackDelegate

2.28 Collections

- Follow the naming standards for Classes.
- **Do not** prefix collections with any letter.
- Always add the suffix “Collection” to the end of the name.
Example: MyCollection

2.29 Attributes

- Follow the naming standards for Classes.
- **Always** add the suffix “Attribute” to custom attribute classes.

The following is an example of a correctly named attribute class.

[Visual Basic]

```
Public Class ObsoleteAttribute
```

[C#]

```
public class ObsoleteAttribute{}
```

2.30 Standard Control Prefixes

2.30.1 Control Objects

Control type	Prefix	Example
Animated button	ani	
Calendar Control	cal	
Check Box	chk	chkPrint
Checked Box List	chkIst	chkIstProducts
Combo Box	cbo	cboTitle
Command button	cmd	
Common dialog	dlg	
Communications	com	
Context Menu	ctx	ctxContactsMenu
Control (generic)	ctl	ctlCurrentControl
Data	dat	
Data Grid	grd	grdProducts
Data list	dbl	
Data repeater	drp	
Date picker	dtp	
Domain up down	dom	domPrimary
Drop-down list box	cbo	
Button	btn	btnOk
Form	frm	
Frame	fra	
Gauge	gau	

Control type	Prefix	Example
Graph	gra	
Group Box	grp	grpDisplayOptions
Help provider	hlp	hlpPrimaryHelp
Horizontal Scroll Bar	hsb	hsbWidth
Image	img	
Image List	lmg	imgIcons
Label	lbl	lblMessage
Link Label	lnk	lnkHomePage
List Box	lst	lstResults
List View	lvw	lvwActiveUsers
Menu Item	mnu	mnuFileOpen
Notify Icon	nfi	nfiTrayIcon
Numeric Up Down	num	numAge
OLE container	ole	
Open File Dialog	opn	opnSelectFile
Panel	pnl	pnlSettings
Progress Bar	prg	prgProgress
Radio Button	opt (for option)	optEnglish
Picture Box	pic	picDiskSpace
Report	rpt	
Rich Text Box	rtf	rtfLayout
Save File Dialog	sav	savSavPicture
Shape	shp	
Slider	sld	
Spin	spn	
Status Bar	sta	staStatus
Tab	tab	tabColorOptions
Text Box	txt	txtAddress
Timer	tmr	tmrAlarm
ToolTip	tip	tipQuickTip
Toolbar	tlb	tlbMain
Track Bar	trk	trkColorDepth
Tree View	tre	treFolders
Splitter	spl	splMainSplitter
Vertical Scroll Bar	vsb	vsbHeight

2.30.2 Database Objects

Database Object type	Prefix	Example
Connection	con	
Data Adapter	da	
Database	db	
DataColumn	dcl	
DataCommand	dcd	
DataReader	drd	
DataRelation	drl	
DataRow	drw	
DataSet	ds	
DataSet Name	dsn	
DataView	dv	
DB Engine	dbe	
Document	doc	
Field	fld	
Function	udf	

Database Object type	Prefix	Example
Group	grp	
Index	idx	
Parameter	prm	
Query Definition	qry	
Record Set	rs	
Relation	rel	
Stored Procedure	usp	
Table		
Table Definition	tbd	
User	usr	
View	vw	
Workspace	wsp	

2.31 ADO.NET Naming Conventions

- Use a descriptive object name that identifies the type of object it is.

2.31 Database Naming Conventions

- Refer to the Information Resource Management: Database Development Standards Document.
- **Database Names:** Consists of a 3 character descriptive acronym for the unit in lowercase followed by a functional description of the database. The first letter of every word and any acronyms must be in uppercase, There must no blanks, underscores or special characters in the name. Examples: finPolicy, repRealNet, psgFuels.
- **Field Names:** Field Names must be a functional description of the field. The first letter of every word and any acronyms must be in uppercase. There must no blanks, underscores or special characters in the name. Examples: LastName, FirstName, OrdersEMAIL.
- **ODBC DSN Names:** SQL Server name that you are connecting to followed by a '.' followed by the name of the database. Examples: RipVanWinkle.Parking, Star.finPolicy.
- **Stored Procedures:** Prefix the stored procedure name with 'usp' in lowercase followed by a functional description of the stored procedure. The first letter of every word and any acronyms must be in uppercase, There must no blanks, underscores or special characters in the name. Example: uspGetOrders.
- **Table Names:** Table Names must be a functional description of the table. The first letter of every word and any acronyms must be in uppercase, There must no blanks, underscores or special characters or in the name. Examples: Employee, Orders, PSGPermits.
- **User Defined Functions:** Prefix the function name with 'udf' in lowercase followed by a functional description of the function. The first letter of every word and any acronyms must be in uppercase. There must no blanks, underscores or special characters in the name. Example: udfCalculateTax.
- **Views Names:** Prefix view names with vw in lowercase followed by a functional description of the view. The first letter of every word and any acronyms must be in uppercase. There must no blanks, underscores or special characters in the name. Example: vwEmployeeTitles

3.0 Comments and Embedded Documentation

3.1 General Rules

Comments must be used to describe intention, algorithmic overview, and/or logical flow. From reading the comments alone, someone other than the author should be able to understand a function's intended behavior and general operation. While there are no minimum comment requirements and certainly some very small routines need no commenting at all, it is hoped that most routines will have comments reflecting the programmer's intent and approach.

- All comments must be written in US English.
- **Always** indent comments to the same level as the code it is related to.
- **Do not** use comments at the end of a line of code. Put comments on a separate line.
- **Use** `///` for comments in C# and `'''` for comments in VB.
- **Always** Insert one space between the comment delimiter (`'''` or `///`) and the comment text.
- **Always** begin the comment text with an uppercase letter.
- **Always** end the comment with a period.
Example: `''' Here is a comment.` or `/// Here is a comment`
- **Do not** create formatted blocks of asterisks or other special characters that surround comments.

3.2 Class Comments

It is recommended to place a comment block before the class declaration with the following detail:

Class Name:	The name of the class.			
Author:	The author's name.			
Description:	What the procedure does (not how).			
Effects: (optional)	List of each affected external variable, control, or file and the effect it has (only if this is not obvious).			
Creation Date:	The date the class was created.			
Copyright:	New York State Office of General Services – All Rights Reserved			
Revision History: *	Revision Number	Who made the last changes	The date modified	Description of the changes made.

3.3 Procedure/Method Comments

It is recommended to place a comment block before the Procedure/Method declaration with the following detail:

Method:	Method Name
Author:	Author's Name
Description:	What the procedure/method does (not how). Update when the procedure/method is modified.
Date	Date of last change
Inputs	The input parameters being used and purpose (optional)
Outputs	The output parameters being used and purpose (optional)
Throws	The exceptions this method throws (optional)

New entries replace the old Author and Date (i.e. only the most recent is listed).

3.4 Special Comments (for development)

Use `TODO` in a comment to flag something that is bogus but works or needs cleaning up. Use `FIXME` to flag something that is bogus and broken. Use your name in uppercase for reminders of things that you need to revisit before the code can be released.

3.5 Header Blocks

Each file must contain a header block. The header block must consist of a `#region/#Region` block containing the following copyright statement and the name of the file.

[VB.NET]

```
#Region Copyright NYS Office of General Service 2008
```

```
' All rights are reserved. Reproduction or transmission in whole or in part, in  
' any form or by any means, electronic, mechanical or otherwise, is prohibited  
' without the prior written consent of the copyright owner.
```

```
' Filename: MyFile.vb
```

```
#EndRegion
```

[C#]

```
#region Copyright NYS Office of General Service 2008
```

```
//  
// All rights are reserved. Reproduction or transmission in whole or in part, in  
// any form or by any means, electronic, mechanical or otherwise, is prohibited  
// without the prior written consent of the copyright owner.
```

```
// Filename: MyFile.cs
```

```
#endregion
```

3.6 Commenting out code vs. removing code

Detect and remove dead code

Code that is never executed is known as dead code. To improve readability and ensure that logic errors are resolved, dead code should be identified, understood, and eliminated.

In some situations, dead code may make software resilient to future changes. An example of this is adding a default case to a switch statement even when all possible switch labels are specified.

It is also permissible to temporarily remove code that may be needed later.

Detect and remove code that has no effect

Statements or expressions that have no effect should be identified and removed from code.

3.8 XML Tags

- All public and protected types, methods, fields, events, delegates, etc. must be documented using XML tags.

Using these tags will allow IntelliSense to provide useful details while using the types. Also, automatic documentation generation tooling relies on these tags.

Section tags define the different sections within the type documentation.

<summary> Short description type or member

<remarks> Describes preconditions and other additional information. type or member

<param> Describes the parameters of a method

<returns> Describes the return value of a method

<exception> Lists the exceptions that a method or property can throw method, even or property

<value> Describes the type of the data a property accepts and/or returns property

<example> Contains examples (code or text) related to a member or a type or member

<seealso> Adds an entry to the *See Also* section type or member

<overloads> Provides a summary for multiple overloads of a method first method in a overload list.

Inline tags can be used within the section tags.

<see> Creates a hyperlink to another member or type

<paramref> Creates a checked reference to a parameter

Markup tags are used to apply special formatting to a part of a section.

<code> Changes the indentation policy for code examples

<c> Changes the font to a fixed-wide font (often used with the <code> tag)

<para> Creates a new paragraph

<list> Creates a bulleted list, numbered list, or a table.

 Bold typeface

<i> Italics typeface

Exception:

In an inheritance hierarchy, do **not** repeat the documentation but use the <see> tag to refer to the base class or interface member.

Exception:

Private and nested classes do not **have** to be documented in this manner.

4.0 Coding Style and Practices

4.1 General Rules

- **Use** only one declaration per line.
- **Use** only one statement per line. Do not place multiple statements on the same line.
- **Use** the line continuation character (_) and use it to promote easy code reading.
- **Consider** using the **With** keyword when faced with a series of calls to one object.
- **Use** the **IsNot** keyword in preference to **Not ... Is Nothing**.

4.2 Indentation and Spacing

4.2.1 Indentation

- **Always** indent to show organizational structure of code.
- **Always** indent continuation lines consistently by using one tab stop.
- **Always** use spaces instead of tab characters. Different applications interpret tabs differently. You must change the settings in Visual Studio .NET for that (Options – Text Editor – All Languages – Tabs).
Set **Tab size** = 4, Set **Indent size** = 4, Select **Insert Spaces**
- **Use** Smart Indenting for VB.NET, C#, C/C++, Visual J#, and XML for all others use Block Indenting.

4.2.2 Spacing

- **Always** insert two blank lines between procedures, methods, and property definitions.
- **Use** two blank lines to separate logical groups of code.
- **Do not** create overly long source code lines. Do not exceed 90 characters on a line. Long lines are hard to read. Many applications, such as printing and difference views, perform poorly with long lines.
- **Do not** mix coding styles within a group of closely related classes or within a module. This coding standard gives you some room in choosing a certain style. Do keep the style consistent within a certain scope. That scope is not rigidly defined here, but is at least as big as a source file.
- The public, protected, and private sections of a class or struct must be declared in that order.
Although C# does not have the same concept of accessibility sections as C++, **do** group them in the given order. However, keep the fields at the top of the class (preferably inside their own #region). The protected internal section goes before the protected section, and the internal section before the private section.
- Write unary, increment, decrement, function call, subscript, and access operators together with their operands.

This concerns the following operators:

unary:	& * + - ~ !
increment and decrement:	-- ++
function call and subscript:	() []
access:	.

- **Use a single space before and after each operator and bracket.**

It is not allowed to add spaces in between these operators and their operands.

It is not allowed to separate a unary operator from its operand with a newline.

Note: this rule does **not** apply to the **binary** versions of the & * + - operators.

Example:

```
a = -- b;           // wrong
a = --c;           // right
a = -b - c;        // right
a = (b1 + b2) +
  (c1 - c2) +
  d - e - f;       // also fine: make it as readable as possible
```

4.3 Procedure Conventions

The following chart shows the preferred order of organization within your classes. Each section should be encased in a region.

Class (shared) variables	First the public class variables, then the protected, and then the private.
Instance variables	First public, then friend, and then private
Methods	These methods should be grouped by functionality rather than by scope or accessibility. For example, a private shared method can be in between two public instance methods. The goal is to make reading and understanding the code easier.

4.4 Region Tags

- Use `#region/#Region` to group non-public members.
- **Always** put all non-public members in a region if a class contains a large number of members, attributes, and/or properties. Preferably, use separate regions to split-up the private, protected, and internal members, and a region to hide all fields. It is also allowed to use the `#region/#Region` construct for separating the smaller auxiliary classes from the main class.

Each region of the code should be encased in the appropriate region tags (ex. `#Region...#End Region`).

Consistency in naming will make all code written by all programmers more readable and therefore reduce costs to our customers in maintenance time.

The general grouping shall be as follows:

- Private Member Variables
- Private Properties
- Private Methods
- Constructors
- Public Properties
- Public Methods

An extra space should be entered in the label at both the beginning and the end in order to improve readability within the IDE.

Examples:

- `#Region " Comments "`
- `#Region " Options "`
- `#Region " Imports "`
- `#Region " Class Constants & Variables "`
- `#Region " Instance Constants & Variables "`
- `#Region " Constructor "`

Note: If there are not many events you can group them all into one section called " Events ," otherwise break them into the type of event that they are.

- `#Region " Key Events "`
- `#Region " Mouse Events "`
- `#Region " Button "`
- `#Region " Shared Methods "`
- `#Region " Methods "`

4.5 Menus

4.5.1 General Rules

The following list outlines some best practices in menu and UI development:

- A single menu bar should be displayed across the top of the main window.
- Menu items should be brief and visible on a single line.
- All menu items should have mnemonics.
- **Use** keyboard shortcuts for frequently used menu items.
- **Use** the same keyboard shortcut if the menu item appears in several menus (contextual and dropdown).
- **Try** to keep to one level of menus with at the most a single level of submenu. If a complex set of sub-sub menus seems necessary then display them in a dialog box instead of making the UI difficult for users to navigate.
- Ellipses should be used on any menu label that requires additional information to complete the operation. i.e. Save As..., users are presented with a file browser dialog to supply the missing filename required to complete the operation. **DO NOT** use ellipses to simply indicate that a dialog or other window will open. i.e. Preferences often opens a dialog box, but the entire effect of that command is to open that dialog box, so you would not use ellipses.
- Group like items into menus and submenus.

Items within a menu can be grouped by related operations and separated from other groupings and operations by a separator bar.

4.5.2 Available and Unavailable Menus and Menu Items

The following defines how to handle available and unavailable menus in your application.

- If the user **CAN** do something to make a menu or menu item available:
 - If a feature is not currently available, but users can do something to make it available. Set **visible** to true and set **enabled** to false. Enabling the item when/if the user's actions make it available.
 - If all the menu items in a menu are currently unavailable do **NOT** make the menu itself unavailable (enable the menu or submenu containing any element that the user could make available). The user should be able to see all functionality they have access to, even if it requires additional actions on their part.
- If the user **CAN NOT** do something to make a menu or menu item available:
 - If there is nothing that a user can do to make a menu item available omit the item entirely by setting **visible** to false.
 - This holds true for sub menus, menu items and contextual menus.

4.5.3 Common Menus

If your application requires these common menus then they should appear in the following order: File, Object, Edit, Format, View, and Help. If needed insert other menus between the View and Help menus.

File Menu – typically the first menu to appear on a menu bar. Place operations to do with the currently active file, project, etc., in this menu. Also any operations that apply to the whole application belong in this menu, such as Exit, Preferences, Print, unless there are enough to warrant their own menu.

Object Menus – are menus that provide actions that users can perform on an object or objects.

Edit Menu – Displays items that enable users to change or edit the contents of whatever your application is manipulating.

Format Menu – Displays items that enable users to change formatting elements within their documents or contents applicable to your application.

View Menu – Provides ways for users to adjust how they can view their application's information.

Help Menu – Provides access to online information about the application. i.e. contents, tutorial, index, search, and About. The About menu should be separated from the help elements in the Help menu by a separator.

Contextual Menus – Sometimes called “pop-up menus,” they provide only menu items that are relevant to the object or region at the location of the mouse cursor. They do not have a menu title, but should be similar in content to the items available on the object, edit and format menus.

Tool Tips (Hover Help) – Whenever possible use tool tips or hover help to assist the user to understand how to interact with the application. These bits of text are displayed when the user's mouse hovers over a UI element or region. They are often used to display complete data that may be partially hidden by UI elements or instructions on how to use a specific button, or format of fields that the user must enter (i.e. password must be 7 – 12 characters long and contain at least one number and one letter).

4.9 Designing Classes, Modules and Procedures

The necessity of designing the structure of code may not be obvious. The structure of the application can make a significant difference in application performance, maintainability and usability of the code.

4.9.1 General Rules

- **Always** use the project settings **Option Explicit = On**, **Option Strict = On**, and set **Option Compare** to Text.
- **Always** assign descriptive names.
- **Avoid** hard coding numbers. When it is needed, use Constants over direct hard-coding of a variable or object.
- **Try** to use Enumerations where possible.
- Minimize the use of module or global level variables.
- Declare variables as close as possible to where it is first used. Use one variable declaration per line.
- **Always** make each variable serve a clearly defined purpose.
- **Try** to minimize variable scope.
- **Use** initializers whenever possible.
- **Always** use the constant names **True** and **False** instead of their underlying values of -1 and 0 when comparing Boolean values.
- **Avoid** hard coding strings. Use resource files.
- **Always** Convert strings to uppercase before comparison.
- **Use** `String.Empty` instead of `""`.
- **Always** use `StringBuilder` when performing string concatenation.
- **Always** use a string's length property to determine whether it's empty.

- **Every** time a standard is not followed, the developer must have a good reason.
- **Avoid** having very large files. If a single file has more than 1000 lines of code, it is a good candidate for refactoring. Split them logically into two or more classes.

4.9.2 Classes and Modules

- Consider using a [Class](#) over a [Module](#) where appropriate.
- [Modules](#) and [Classes](#) should be used to organize related procedures (strong cohesion)
- Do not have more than one class in a single file.

4.9.3 Procedures (Methods, Functions and Sub-Routines)

- **Try** to make procedures as specialized and self-contained as possible. This will allow for reusable and maintainable code.
- **Always** pass in the data or object(s) that the procedure needs to perform the task instead of relying on global or module level variables inside of the procedure.
- **Do not** pass parameters by reference if possible.
- Procedures should only have one exit point.
- **Always** use the Return statement to return values from a function.
- **Always** store the returned value (even if it is not used) when calling a function.
- **Always** clearly define a procedures scope using the appropriate [Access Modifier](#) (**See Section 1.3**).
- **Always** validate procedure parameters. Validating parameters is the first task that should be performed by the procedure.
- **Use** enumeration parameter types where possible. Do not use numbers or strings to indicate discrete values. Enumerations help to reduce the risk of data entry errors in coding.
- **Consider** breaking a procedure down into smaller constituent components if the procedure calls approximately 20 other procedures to complete it's processing,
- A method should have no more than 25 lines of code. If it is larger that it should be re-factored into separate methods.
- A method should do only one job. Do not combine more than one job in a single method even if those jobs are small.
- **Avoid** using member variables. Declare local variables instead and pass it to other methods instead of sharing a member variable between methods.
- **Do not** make the member variables public or protected. Keep them private and expose public/protected Properties.
- The event handler should not contain the code to perform the required action. Rather call another method from the event handler.
- **Do not** programmatically click a button to execute the same action you have written in the button click event. Rather, call the same method which is called by the button click event handler.
- **Never** hardcode a path or drive name in code. Get the application path programmatically and use relative path.
- **Never** assume that your code will run from drive "C:". You may never know, some users may run it from network or from a "Z:".
- In the application start up, do some kind of "self check" and ensure all required files and dependencies are available in the expected locations. Check for database connection in start up, if required. Give a friendly message to the user in case of any problems.
- If the required configuration file is not found, application should be able to create one with default values.

- If a wrong value found in the configuration file, application should throw an error or give a message and also should tell the user what are the correct values.
- Have your own templates for each of the file types in Visual Studio. You can include your company name, copy right information etc in the template. You can view or edit the Visual Studio file templates in the folder `C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\ItemTemplatesCache\CSharp\1033`. (This folder has the templates for C#, but you can easily find the corresponding folders or any other language)
- **Avoid** public methods and properties, unless they really need to be accessed from outside the class. Use “internal” if they are accessed only within the same assembly.
- **Avoid** passing too many parameters to a method. If you have more than 4-5 parameters, it is a good candidate to define a class or structure.
- If you have a method returning a collection, return an empty collection instead of null, if you have no data to return. For example, if you have a method returning an ArrayList, always return a valid ArrayList. If you have no items to return, then return a valid ArrayList with 0 items. This will make it easy for the calling application to just check for the “count” rather than doing an additional check for “null”.
- **Use** the AssemblyInfo file to fill information like version number, description, company name, copyright notice etc.

4.10 Object Life Cycle

4.10.1 General Rules

- Declare and initialize variables close to where they are used.
- If possible, initialize variables at the point of declaration.

Avoid the C style where all variables have to be defined at the beginning of a block, but rather define and initialize each variable at the point where it is needed.

4.10.2 Constant Values

- **Use** a const field to define constant values. Making it `const` ensures that memory is allocated for that item only once.

[C#]

```
private const int maxUsers = 100;
```

Exception: If the value of a constant field must be calculated at run-time (in the static constructor), use a `static readonly` field instead (**See Section 4.10.3**).

4.10.3 Public Static Read-Only Fields

- **Use** a public static read-only field to define predefined object instances.

For example, consider a Color class/struct that expresses a certain color internally as red, green, and blue components, and this class has a constructor taking a numeric value, then this class may expose several predefined colors like this.

[C#]

```
public struct Color
{
    public static readonly Color Red = new Color(0xFF0000);
    public static readonly Color Black = new Color(0x000000);
    public static readonly Color White = new Color(0xFFFFFFFF);
    public Color(int rgb)
    {
        // implementation
    }
}
```

4.10.4 Reference Fields

- Set a reference field to null to tell the GC that the object is no longer needed.

Setting reference fields to null may improve memory usage because the object involved will be unreferenced from that point on, allowing the GC to clean-up the object much earlier. Please note that this recommendation should **not** be followed for a variable that is about to go out of scope.

4.10.5 Shadowing

- **Do not** 'shadow' a name in an outer scope.

Repeating a name that already occurs in an outer scope is seldom intended and may be surprising in maintenance, although the behavior is well-defined.

```
[C#]
int foo = something;
if (whatever)
{
    double foo = 12.34; // do not re-use this name
}
```

An exception is made for the case where a parameter of a method has the same name as a field, usually causing a construction like `this.foo = foo` .

4.10.6 Destructors

- **Avoid** implementing a destructor.
- If a destructor is required, follow the “Classes using unmanaged or expensive resources” guidelines. **See Section 4.10.7**

The use of destructors in C# is demoted since it introduces a severe performance penalty due to way the Garbage Collector (GC) works. It is also a bad design pattern to clean up any resources in the destructor since you cannot predict at which time the destructor is called (in other words, it is non-deterministic).

Notice that C# destructors are not really destructors as in C++. They are just a C# compiler feature to represent CLR Finalizers.

- **Use GC.SuppressFinalize** if a destructor is needed.

If a destructor is needed to verify that a user has called certain cleanup methods such as **Close()** on a **IpcPeer** object, call **GC.SuppressFinalize** in the **Close()** method. This ensures that the destructor is ignored if the user is properly using the class. The following snippet illustrates this pattern.

```
[C#]
public class IpcPeer
{
    bool connected = false;
    public void Connect()
    {
        // Do some work and then change the state of this object.
        connected = true;
    }
    public void Close()
    {
        // Close the connection, change the state, and instruct the GC
        // not to call the destructor.
        connected = false;
        GC.SuppressFinalize(this);
    }
    ~IpcPeer()
    {
        // If the destructor is called, then Close() was not called.
        if (connected)
        {
            // Warning! User has not called Close(). Notice that you can't
            // call Close() from here because the objects involved may
            // have already been garbage collected (See Section 4.10.7).
        }
    }
}
```

4.10.7 Classes using unmanaged or expensive resources

- Implement IDisposable if a class uses unmanaged or expensive resources.

If a class uses unmanaged resources such as objects returned by C/C++ DLLs, or expensive resources that must be disposed of as soon as possible, you must implement the IDisposable interface to allow class users to explicitly release such resources.

The follow code snippet shows the pattern to use for such scenarios.

```
[C#]
public class ResourceHolder : IDisposable
{
    ///<summary>
    ///Implementation of the IDisposable interface
    ///</summary>
    public void Dispose()
    {
        // Call internal Dispose(bool)
        Dispose(true);
        // Prevent the destructor from being called
        GC.SuppressFinalize(this);
    }
    ///<summary>
    /// Central method for cleaning up resources
    ///</summary>
    protected virtual void Dispose3(bool explicit)
    {
        // If explicit is true, then this method was called through the
        // public Dispose()
        if (explicit)
        {
            // Release or cleanup managed resources
        }
        // Always release or cleanup (any) unmanaged resources
    }
    ~ResourceHolder()
    {
        // Since other managed objects are disposed automatically, we
        // should not try to dispose any managed resources (See Section 4.10.8).
        // We therefore pass false to Dispose()
        Dispose(false);
    }
}
```

³ Please note that this method could have any other name, e.g. InternalDispose. It has **no** relation to the parameterless Dispose() method of IDisposable.

If another class derives from this class, then this class should only override the `Dispose(bool)` method of the base class. It should not implement `IDisposable` itself, nor provide a destructor. The base class's 'destructor' is automatically called.

```
[C#]
public class DerivedResourceHolder : ResourceHolder
{
    protected override void Dispose(bool explicit)
    {
        if (explicit)
        {
            // Release or cleanup managed resources of this derived
            // class only.
        }
        // Always release or cleanup (any) unmanaged resources.
        // Call Dispose on our base class.
        base.Dispose(explicit);
    }
}
```

4.10.8 Reference Type Members in the Destructor

- **Do not** access any reference type members in the destructor. When the destructor is called by the GC, it is very possible that some or all of the objects referenced by class members are already garbage collected, so dereferencing those objects may cause exceptions to be thrown.

Only value type members can be accessed (since they live on the stack).

4.10.9 Returning a Copy of a Reference Type or Array

- **Always** document when a member returns a copy of a reference type or array. By default, all members that need to return an internal object or an array of objects will return a reference to that object or array. In some cases, it is safer to return a copy of an object or an array of objects. In such case, **always** clearly document this in the specification.

4.11 Control Flow

4.11.1 Loop Variables

- **Do not** change a loop variable inside a for loop block. Updating the loop variable within the loop body is generally considered confusing, even more so if the loop variable is modified in more than one place. This rule also applies to for-each loops.
- **Always** update loop variables close to where the loop condition is specified. This makes understanding the loop much easier.

4.11.2 Flow Control Primitives (if, else, while, for, do, switch)

- **All** flow control primitives (if, else, while, for, do, switch) must be followed by a block, even if it is empty. Please note that this also avoids possible confusion in statements of the form:

Example:

```
[C#]
if (b1) if (b2) Foo(); else Bar(); // which 'if' goes with the 'else'?
```

4.11.3 Switch Statements

- **All** switch statements must have a default label as the last case label. A comment such as “*no action*” is recommended where this is the explicit intention. If the default case should be unreachable, an assertion to this effect is recommended.

If the default label is always the last one, it is easy to locate.

4.11.4 Else If Sub-Statement

Always use an *else* sub-statement when using an *else if* sub-statement of an *if* statement. The intention of this rule, which applies to else-if constructs, is the **same as in Section 4.11.3**. Consider the following example.

```
[C#]
void Foo(string answer)
{
    if ("no" == answer)
    {
        Console.WriteLine("You answered with No");
    }
    else if ("yes" == answer)
    {
        Console.WriteLine("You answered with Yes");
    }
    else
    {
        // This block is required, even though you might not want any
        // answers other than "yes" and "no".
    }
}
```

4.11.5 Return Statements.

- **Avoid** multiple or conditional return statements. *One entry, one exit* is a sound principle and keeps control flow simple. However, if some cases, such as when preconditions are checked, it may be good practice to exit a method immediately when a certain precondition is not met.

4.11.6 Comparisons

- **Do not** make explicit comparisons to true or false. It is usually bad style to compare a boolean-type expression to true or false.

Example:

```
[C#]
while (condition == false)           // wrong; bad style
while (condition != true)           // also wrong
while (((condition == true) == true) == true) // where do you stop?
while (condition)                   // OK
```

4.11.7 Accessing modified objects

- **Do not** access a modified object more than once in an expression. The evaluation order of sub-expressions within an expression is defined in C#, in contrast to C or C++, but such code is hard to understand.

Example:

[C#]

```
v[i] = ++c;    // right
v[i] = ++i;    // wrong: is v[i] or v[++i] being assigned to?
i = i + 1;     // right
i = ++i + 1;   // wrong and useless; i += 2 would be more clear
```

4.11.8 Simple Assignment or Initialization

- **Do not** use selection statements (if, switch) instead of a simple assignment or initialization. Express your intentions directly.

Example:

[C#]

Rather than

```
bool pos;
if (val > 0)
{
    pos = true;
}
else
{
    pos = false;
}
```

or (slightly better)

```
bool pos = (val > 0) ? true : false;
```

write

```
bool pos;
pos = (val > 0);    // single assignment
```

or even better

```
bool pos = (val > 0);    // initialization
```

4.12 Object Oriented Programming

4.12.1 General Rules

- Declare all fields (data members) private.

Exceptions to this rule are static read-only fields and const fields, which may have any accessibility deemed appropriate (**See Section 4.10.3**).

- Provide a default private constructor if there are only static methods and properties on a class. Instantiating such a class would be useless.
- Explicitly define a protected constructor on an abstract base class.

Of course an abstract class cannot be instantiated, so a public constructor should be harmless. However, MSDN states:

Many compilers will insert a public or protected constructor if you do not. Therefore, for better documentation and readability of your source code, you should explicitly define a protected constructor on all abstract classes.

- Selection statements (if-else and switch) should be used when the control flow depends on an object's value; dynamic binding should be used when the control flow depends on the object's type.

Please note that it is usually a design error to write a selection statement that queries the type of an object (keywords `typeof`, `is`).

Exception:

Using a selection statement to determine if some object implements one or more optional interfaces is a valid construct though.

- All variants of an overloaded method must be used for the same purpose and have similar behavior. Doing otherwise is against the *Principle of Least Surprise*.

4.12.2 Overriding a Method

- If you must provide the ability to override a method, make only the most complete overload virtual and define the other operations in terms of it.

Using the pattern illustrated below requires a derived class to only override the virtual method. Since all the other methods are implemented by calling the most complete overload, they will automatically use the new implementation provided by the derived class.

```
[C#]
public class MultipleOverrideDemo
{
    private string someText;
    public MultipleOverrideDemo(string s)
    {
        this.someText = s;
    }
    public int IndexOf(string s)
    {
        return IndexOf(s, 0);
    }
    public int IndexOf(string s, int startIndex)
    {
        return IndexOf(s, startIndex, someText.Length - startIndex );
    }
    public virtual int IndexOf(string s, int startIndex, int count)
    {
        return someText.IndexOf(s, startIndex, count);
    }
}
```

An even better approach, **not** required by this coding standard, is to refrain from making virtual methods public, but to give them protected accessibility, changing the sample above into:

```
[C#]
public class MultipleOverrideDemo
{
    // same as above ...
    public int IndexOf(string s, int startIndex, int count)
    {
        return InternalIndexOf(s, startIndex, count);
    }
    protected virtual int InternalIndexOf(string s, int startIndex, int count)
    {
        return someText.IndexOf(s, startIndex, count);
    }
}
```

4.12.3 Specifying Methods and Classes

- Specify methods using Pre-conditions, Post-conditions, Exceptions; Specify classes using Invariants
You can use **Debug.Assert** to ensure that pre- and post-conditions are only checked in debug builds. In release builds, this method does not result in any code.

4.12.4 Describe pre-conditions, post-conditions, exceptions, and class invariants.

- Use C# to describe pre-conditions, post-conditions, exceptions, and class invariants. Compile-able preconditions etc. are testable.
The exact form (e.g. assertions, special DbC functions such as *require* and *ensure*) is not discussed here. However, a non-testable (text only) precondition is better than a missing one.

4.12.5 Referencing an Object of a Derived Class

- It shall be possible to use a reference to an object of a derived class where ever a reference to that object's base class object is used.
This rule is known as the *Liskov Substitution Principle*, often abbreviated to *LSP*. Please note that an interface is also regarded as a base class in this context.

4.12.6 Overloading 'modifying' operators on a class type

- **Do not** overload any 'modifying' operators on a class type.
In this context the 'modifying' operators are those that have a corresponding assignment operator, i.e. the non-unary versions of +, -, *, /, %, &, |, ^, << and >>. There is very little literature regarding operator overloading in C#. Therefore it is wise to approach this feature with some caution.
Overloading operators on a struct type is good practice, since it is a value type. The class is a reference type and users will probably expect reference semantics, which are not provided by most operators.
Consider a class **Foo** with an overloaded operator **+(int)**, and thus an implicitly overloaded operator **+=(int)**. If we define the function **AddTwenty** as follows:

```
[C#]
public static void AddTwenty (Foo f)
{
    f += 20;
}
```

Then this function has **no** net effect:

```
[C#]
{
    Foo bar = new Foo(5);
    AddTwenty (bar);
    // note that 'bar' is unchanged
    // the Foo object with value 25 is on its way to the GC...
}
```

The exception to this recommendation is a class type that has complete value semantics, like System.String.

4.12.7 Implementation of an Overloaded Operator

- Do not modify the value of any of the operands in the implementation of an overloaded operator.

This rule can be found in a non-normative clause of **ECMA-334, Section 17.9.1**. Breaking this rule gives counter-intuitive results.

4.12.8 The operator==(), Equals method and GetHashCode()

- If you implement one of operator==(), the Equals method or GetHashCode(), implement all three.
- Also override this trio when you implement the IComparable interface.
- Do consider implementing all relational operators (!=, <, <=, >, >=) if you implement any.
- If your Equals method can throw an exception, this may cause problems if objects of that type are put into a container. Do consider to return false for a null argument.

The MSDN guidelines recommend returning false rather than throwing an exception when two incomparable objects, say the proverbial apples and oranges, are compared. Since this approach sacrifices the last remnants of type-safety, this recommendation has been weakened.

Exceptions:

In very rare cases it can be meaningful to override GetHashCode() without implementing the other two. If you implement the Equals method on a *reference* type you do not **have** to implement operator==().

4.12.9 Structs and Value Semantics

- **Use** a struct when value semantics are desired.

More precisely, a struct should be considered for types that meet any of the following criteria:

Act like primitive types.

Have an instance size under $\lll 16$ bytes.

Are immutable.

Value semantics are desirable.

Remember that a struct cannot be derived from.

4.12.10 Properties

- **Allow** properties to be set in any order. Properties should be stateless with respect to other properties, i.e. there should not be an observable difference between first setting property A and then B and its reverse.
- **Use** a property rather than a method when the member is a logical data member.

4.12.11 Methods vs. Properties

- Use a method rather than a property when this is more appropriate.

In some cases a method is better than a property:

- The operation is a conversion, such as `Object.ToString`.
- The operation is expensive enough that you want to communicate to the user that they should consider caching the result.
- Obtaining a property value using the get accessor would have an observable side effect.
- Calling the member twice in succession produces different results.
- The order of execution is important (**See Section 4.12.10**).
- The member is static but returns a value that can be changed.
- The member returns a copy of an internal array or other reference type.
- Only a set accessor would be supplied. Write-only properties tend to be confusing.

4.12.17 Constructors

- Do not create a constructor that does not yield a fully initialized object.

Only create constructors that construct objects that are fully initialized. There shall be no need to set additional properties. A private constructor is exempt from this rule.

4.12.18 Operation Results

- Always check the result of an as operation.

If you use `as` to obtain a certain interface reference from an object, always ensure that this operation does not return null. Failure to do so may cause a `NullReferenceException` at a later stage if the object did not implement that interface.

4.12.19 Interfaces

- **Use** explicit interface implementation only to prevent name clashing or to support optional interfaces.

When you use explicit interface implementation, then the methods implemented by the class involved will not be visible through the class interface. To access those methods, you must first cast the class object to the requested interface.

It is recommended to use explicit interface implementation only:

When you want to prevent name clashing. This can happen when multiple interfaces must be supported which have equally named methods, or when an existing class must support a new interface in which the interface has a member which name clashes with a member of the class.

When you want to support several optional interfaces (e.g. IEnumerator, IComparer, etc) and you do not want to clutter your class interface with their members.

Consider the following example.

```
[C#]
public interface IFoo1
{
    void Foo()
}
public interface IFoo2
{
    void Foo()
}
public class FooClass : IFoo1, IFoo2
{
    // This Foo is only accessible by explicitly casting to IFoo1
    void IFoo1.Foo() { ... }
    // This Foo is only accessible by explicitly casting to IFoo2
    void IFoo2.Foo() { ... }
}
```

4.13 Various Data Types

4.13.1 Enumerations

- **Use** an enum to strongly type parameters, properties, and return types.

This enhances clarity and type-safety. Try to avoid casting between enumerated types and integral types.

Exception: In some cases, such as when databases or MIT interfaces that store values as ints are involved, using enums will result in an unacceptable amount of casting. In that case, it is better to use a const int construction.

- **Use** the default type Int32 as the underlying type of an enum unless there is a reason to use Int64.

If the enum represents flags and there are currently more than 32 flags, or the enum might grow to that many flags in the future, use Int64.

- Do not use any other underlying type because the Operating System will try to align an enum on 32-bit or 64-bit boundaries (depending on the hardware platform). Using an 8-bit or 16-bit type may result in a performance loss.

4.13.2 The [Flags] attribute on an enumeration

Use the [Flags] attribute on an enum if a bitwise operation is to be performed on the numeric values.

Use an enum with the flags attribute only if the value can be completely expressed as a set of bit flags. Do not use an enum for open sets (such as the operating system version). Use a plural name for such an enum (**See Section 2.13**).

Usage and effect of this attribute are not mentioned in **ECMA-334** and not at all clearly in the online documentation, so the benefits of following this recommendation are not obvious. The intended use appears to be:

[C#]

[Flags]

```
public enum AccessPrivileges
{
    Read = 0x1,
    Write = 0x2,
    Append = 0x4,
    Delete = 0x8,
    All = Read | Write | Append | Delete
}
```

4.13.3 “Magic Numbers”

- Do not use “magic numbers”.

Do not use literal values, either numeric or strings, in your code other than to define symbolic constants. Use the following pattern to define constants:

[C#]

```
public class Whatever
{
    public static readonly Color PapayaWhip = new Color(0xFFEFD5);
    public const int MaxNumberOfWheels = 18;
}
```

There are exceptions: the values 0, 1 and null can nearly always be used safely. Very often the values 2 and -1 are OK as well. Strings intended for logging or tracing are exempt from this rule. Literals are allowed when their meaning is clear from the context, and not subject to future changes.

[C#]

```
mean = (a + b) / 2; // okay
WaitMilliseconds(waitTimeInSeconds * 1000); // clear enough
```

If the value of one constant depends on the value of another, do attempt to make this explicit in the code, so do **not** write the following

[C#]

```
public class SomeSpecialContainer
{
    public const int MaxItems = 32;
    public const int HighWaterMark = 24; // at 75%
}
```

but rather write

[C#]

```
public class SomeSpecialContainer
{
    public const int MaxItems = 32;
    public const int HighWaterMark = 3 * MaxItems / 4; // at 75%
}
```

Please note that an enum can often be used for certain types of symbolic constants.

4.13.4 Floating Point Values

- Floating point values must not be compared using either the == or != operators.

Most floating point values have no exact binary representation and have a limited precision.

Exception:

When a floating point variable is explicitly initialized with a value such as 1.0 or 0.0, and then checked for a change at a later stage.

4.13.5 Casting Types

- **Do not** cast types where a loss of precision is possible.

Example:

Do not cast a long (64-bit) to an int (32-bit), unless you can guarantee that the value of the long is small enough to fit in the int.

- Only implement casts that operate on the complete object.

In other words, do not cast one type to another using a member of the source type. For example, a Button class has a string property Name. It is valid to cast the Button to the Control (since Button **is a** Control), but it is not valid to cast the Button to a string by returning the value of the Name property.

- Do not generate a semantically different value with a cast.

For example, it is appropriate to convert a Time or TimeSpan into an Int32. The Int32 still represents the time or duration. It does not, however, make sense to convert a file name string such as c:\mybitmap.gif into a Bitmap object.

4.14 Delegates and Events

4.14.1 Object State

- **Do not** make assumptions on the object's state after raising an event.

Prepare for any changes to the current object's state while executing an event handler. The event handler may have called other methods or properties that changed the object's state (e.g. it may have disposed objects referenced through a field).

4.14.2 Threads

- **Always** document from which thread an event handler is called.

Some classes create a dedicated thread or use the Thread Pool to perform some work, and then raise an event. The consequence of that is that an event handler is executed from another thread than the main thread. For such an event, the event handler must synchronize (ensure thread-safety) access to shared data (e.g. instance members).

4.14.3 Raising Events

- Raise events through a protected virtual method.

If a derived class wants to intercept an event, it can override such a virtual method, do its own work, and then decide whether or not to call the base class version. Since the derived class may decide not to call the base class method, ensure that it does not do any work required for the base class to function properly.

Name this method `OnEventName`, where `EventName` should be replaced with the name of the event. Notice that an event handler uses the same naming scheme but has a different signature. The following snippet (most parts left out for brevity) illustrates the difference between the two.

[C#]

```
///
```

4.14.4 Sender / Arguments

- **Use** the sender/arguments signature for event handlers.

The goal of this recommendation is to have a consistent signature for all event handlers. In general, the event handler's signature should look like the following:

Example: `public delegate void MyEventHandler(object sender, EventArgs arguments)`

Using the base class as the sender type allows derived classes to reuse the same event handler.

The same applies to the arguments parameter. It is recommended to derive from the .NET Framework's `EventArgs` class and add your own event data. Using such a class prevents cluttering the event handler's signature, allows extending the event data without breaking any existing users, and can accommodate multiple return values (instead of using reference fields). Moreover, all event data should be exposed through properties, because that allows for verification and preventing access to data that is not always valid in all occurrences of a certain event.

4.14.5 Add / Remove Accessors

- Implement add/remove accessors if the number of handlers for an event must be limited.

If you implement the add and remove accessors of an event, then the CLR will call those accessors when an event handler is added or removed. This allows limiting the number of allowed event handlers, or to check for certain preconditions.

4.14.6 Property-Changed Events

- **Consider** providing property-changed events.

Consider providing events that are raised when certain properties are changed. Such an event should be named `PropertyChanged`, where `Property` should be replaced with the name of the property with which this event is associated.

4.14.7 Interface vs. Delegate.

- **Consider** an interface instead of a delegate.

If you provide a method as the target for a delegate, the compiler will only ensure that the method signature matches the delegate's signature.

This means that if you have two classes providing a delegate with the same signature and the same name, and each class has a method as a target for that delegate, it is possible to provide the method of the first class as a target for the delegate in the other class, even though they might not be related at all.

Therefore, it is sometimes better to use interfaces. The compiler will ensure that you cannot accidentally provide a class implementing a certain interface to a method that accepts another interface that happens to have the same name.

4.15 Exception Handling

4.15.1 General Rules

The following rules outline the guidelines for raising and handling errors:

- **Always** log detailed errors to an “ErrorLog” table in the SQL database that is associated with the application.
- Make sure you have a good logging class which can be configured to log errors, warning or traces. If you configure to log errors, it should only log errors. But if you configure to log traces, it should record all (errors, warnings and trace). Your log class should be written such a way that in future you can change it easily to log to Windows Event Log, SQL Server, or Email to administrator or to a File etc without any change in any other part of the application. Use the log class extensively throughout the code to record errors, warning and even trace messages that can help you trouble shoot a problem.
- All code paths that result in an exception should provide a method to check for success without throwing an exception. This might not always be possible, but the goal is that under normal execution no exceptions should be thrown.

Example: to avoid a FileNotFoundException you can call File.Exists.

- Use the common constructors shown in the following code example when creating exception classes.
- In most cases, use the predefined exception types. Only define new exception types for programmatic scenarios, where you expect users of your class library to catch exceptions of this new type and perform a programmatic action based on the exception type itself. This is in lieu of parsing the exception string, which would negatively impact performance and maintenance.
Example: it makes sense to define a FileNotFoundException because the developer might decide to create the missing file. However, a IOException is not something that would typically be handled specifically in code.
- You should always explicitly check for errors rather than waiting for exceptions to occur. On the other hand, you should always use exception handlers while you communicate with external systems like network, hardware devices etc. Such systems are subject to failure anytime and error checking is not usually reliable. In those cases, you should use exception handlers and try to recover from error.
- Write your own custom user-defined exception classes if required in your application. Do not derive your custom user-defined exceptions from the Exception base class. Instead, For most applications, derive custom user-defined exceptions from the ApplicationException class.
- The method you choose depends on how often you expect the event to occur. If the event is truly exceptional and is an error (such as an unexpected end-of-file), using exception handling is better because less code is executed in the normal case. If the event happens routinely, using the programmatic method to check for errors is better. In this case, if an exception occurs, the exception will take longer to handle.
- In most cases, use the predefined exceptions types. Define new exception types only for programmatic scenarios. Introduce a new exception class to enable a programmer to take a different action in code based on the exception class.
- Provide **Exception** properties for programmatic access. Include extra information in an exception (in addition to the description string) only when there is a programmatic scenario where the additional information is useful.
- Throw an InvalidOperationException if a property set or method call is not appropriate given the object's current state.
- Throw an ArgumentException or a class derived from ArgumentException if invalid parameters are passed.
- The stack trace begins at the statement where the exception is thrown and ends at the catch statement that catches the exception. Be aware of this fact when deciding where to place a throw statement.

- Use exception builder methods. It is common for a class to throw the same exception from different places in its implementation. To avoid excessive code, use helper methods that create the exception and return it. For example:

[VB.NET]

```
Class File
    Private fileName As String
    Public Function Read(bytes As Integer) As Byte()
        If Not ReadFile(handle, bytes) Then
            Throw NewFileIOException()
        End If
    End Function 'Read
    Function NewFileIOException() As FileException
        Dim description As String = __unknown ' Build localized string, including fileName.
        Return New FileException(description)
    End Function 'NewFileIOException
End Class 'File
```

- Alternatively, use the exception's constructor to build the exception. This is more appropriate for global exception classes, such as **ArgumentException**.
- Throw exceptions instead of returning an error code or HRESULT.
- Clean up intermediate results when throwing an exception. Callers should be able assume that there are no side effects when an exception is thrown from a method.

4.15.2 Try/Catch Blocks

- Use try/finally blocks around code that can potentially generate an exception and centralize your catch statements in one location. In this way, the try statement generates the exception, the finally statement closes or de-allocates resources, and the catch statement handles the exception from a central location.
- Always order exceptions in catch blocks from the most specific to the least specific. This technique handles the specific exception before it is passed to a more general catch block.
- Do not write try-catch in all your methods. Use it only if there is a possibility that a specific exception may occur and it cannot be prevented by any other means.
- Do not write very large try-catch blocks. If required, write separate try-catch for each task you perform and enclose only the specific piece of code inside the try-catch. This will help you find which piece of code generated the exception and you can give specific error message to the user.
- Know when to set up a try/catch block. Never do a 'catch exception and do nothing'. If you hide an exception, you will never know if the exception happened or not. Lot of developers uses this handy method to ignore non significant errors. You should always try to avoid using exception handling by programmatically checking all the error conditions that are likely to occur. In any case, catching an exception and doing nothing is not allowed. In the worst case, you should log the exception and proceed.

Example: Using an **if** statement to check whether a connection is closed. You can use this method instead of throwing an exception if the connection is not closed.

[VB.NET]

```
If conn.State <> ConnectionState.Closed Then
    conn.Close()
End If
```

Example: An exception is thrown if the connection is not closed or was never open (You cannot close an unopened connection).

[VB.NET]

```
Try
    conn.Close()
Catch ex As InvalidOperationException
    'Do something with the error or ignore it.
End Try
```

4.15.3 Throw the most specific exception possible.

- Always catch only the specific exception, not generic exception.
- Do not throw a generic exception if a more specific one is available (**See Section 6.3**).
- There is no need to catch the general exception in all your methods. Leave it open and let the application crash. This will help you find most of the errors during development cycle. You can have an application level (thread level) error handler where you can handle all general exceptions.
- In case of exceptions, give a friendly message to the user before closing the application, or allowing the user to 'ignore and proceed'. Log the actual error with all possible details about the error, including the time it occurred, method and class name etc.

4.15.4 Throw informational exceptions.

- When you instantiate a new exception, set its Message property to a descriptive message that will help the caller to diagnose the problem.

Example: If an argument was incorrect, indicate which argument was the cause of the problem. Also mention the name (if available) of the object involved.

Also, if you design a new exception class, note that it is possible to add custom properties that can provide additional details to the caller.

- Error messages should help the user to solve the problem. Never give error messages like "Error in Application", "There is an error" etc. Instead give specific messages like "Failed to update database. Please make sure the login id and password are correct."
- When displaying error messages, in addition to telling what is wrong, the message should also tell what the user should do to solve the problem. Instead of message like "Failed to update database.", suggest what the user should do: "Failed to update database. Please make sure the login id and password are correct."
- Show short and friendly message to the user. But log the actual error with all possible information. This will help a lot in diagnosing problems.
- Include a localized description string in every exception. When the user sees an error message, it is derived from the description string of the exception that was thrown, rather than from the exception class.
- Use grammatically correct error messages, including ending punctuation. Each sentence in a description string of an exception should end in a period.

4.15.5 Always log that an exception is thrown.

- Logging ensures that if the caller catches your exception and discards it, traces of this exception can be recovered at a later stage.

4.15.6 Only throw exceptions in exceptional situations.

- Do not throw exceptions in situations that are normal or expected (e.g. end-of-file). Use return values or status enumerations instead. In general, try to design classes that do not throw exceptions in the normal flow of control.

However, **do** throw exceptions that a user is not allowed to catch when a situation occurs that may indicate a design error in the way your class is used.

- Return null for extremely common error cases.

Example:

File.Open returns null if the file is not found, but throws an exception if the file is locked.

- Design classes so that an exception is never thrown in normal use. For example, a **FileStream** class exposes another way of determining whether the end of the file has been reached. This avoids the exception that is thrown if you read past the end of the file. The following example shows how to read to the end of the file.

[VB.NET]

```
Class FileRead
  Sub Open()
    Dim stream As FileStream = File.Open("myfile.txt", FileMode.Open)
    Dim b As Byte
    ' ReadByte returns -1 at EOF.
    While b = stream.ReadByte() <> True
      ' Do something.
    End While
  End Sub
End Class
```

4.15.6 Do not throw exceptions from inside destructors.

- When you call an exception from inside a destructor, the CLR will stop executing the destructor, and pass the exception to the base class destructor (if any). If there is no base class, then the destructor is discarded.

4.15.7 Only re-throw exceptions when you want to specialize the exception.

- Only catch and re-throw exceptions if you want to add additional information and/or change the type of the exception into a more specific exception. In the latter case, set the `InnerException` property of the new exception to the caught exception.
- When you re-throw an exception, use the `throw` statement without specifying the original exception. This way, the original call stack is preserved.

Good:

```
catch
{
  // do whatever you want to handle the exception

  throw;
}
```

Not Good:

```
catch (Exception ex)
{
  // do whatever you want to handle the exception

  throw ex;
}
```

4.15.8 List the explicit exceptions

- List the explicit exceptions a method or property can throw.
- Describe the recoverable exceptions using the <exception> tag
- Explicit exceptions are the ones that a method or property explicitly throws from its implementation and which users are allowed to catch. Exceptions thrown by .NET framework classes and methods used by this implementation do not have to be listed here.

4.15.9 Allowing callers to prevent exceptions

Allow callers to prevent exceptions by providing a method or property that returns the object's state.

Example: Consider a communication layer that will throw an `InvalidOperationException` when an attempt is made to call `Send()` when no connection is available. To allow preventing such a situation, provide a property such as `Connected` to allow the caller to determine if a connection is available before attempting an operation.

4.15.10 Use standard exceptions.

The .NET framework already provides a set of common exceptions. The runtime has a base set of exceptions deriving from **SystemException** that it throws when executing individual instructions. The following table hierarchically lists the standard exceptions provided by the runtime and the conditions under which you should create a derived class.

Exception type	Base type	Description	Example
Exception	Object	Base class for all exceptions.	None (use a derived class of this exception).
SystemException	Exception	Base class for all runtime-generated errors.	None (use a derived class of this exception).
ApplicationException	Exception	General Application error has occurred that does not fit in the other more specific exception classes.	
IndexOutOfRangeException	SystemException	Thrown by the runtime only when an array is indexed improperly. Indexing an array or index-able collection outside its valid range.	Indexing an array outside its valid range: <code>arr[arr.Length+1]</code>
NullReferenceException	SystemException	Thrown by the runtime only when a null object is referenced.	<code>object o = null;</code> <code>o.ToString();</code>
InvalidOperationException	SystemException	Thrown by methods when in an invalid state. An action is performed which is not valid considering the object's current state.	Calling <code>Enumerator.GetNext()</code> after removing an <code>Item</code> from the underlying collection.
ArgumentException	SystemException	Base class for all argument exceptions. An incorrect argument is supplied.	None (use a derived class of this exception).
ArgumentNullException	ArgumentException	Thrown by methods that do not allow an argument to be null. A null reference is supplied as a method's parameter that does not allow null.	<code>String s = null;</code> <code>"Calculate".IndexOf (s);</code>

Exception type	Base type	Description	Example
ArgumentOutOfRangeException	ArgumentException	Thrown by methods that verify that arguments are in a given range. An argument is not within the required range.	String s = "string"; s.Chars[9];
ExternalException	SystemException	Base class for exceptions that occur or are targeted at environments outside the runtime.	None (use a derived class of this exception).
ComException	ExternalException	Exception encapsulating COM HRESULT information.	Used in COM interop.
SEHException	ExternalException	Exception encapsulating Win32 structured exception handling information.	Used in unmanaged code interop.
NotSupportedException	SystemException	An action is performed which is may be valid in the future, but is not supported.	

4.15.11 Only catch the exceptions explicitly mentioned in the documentation.

- Moreover, do not catch the base class Exception or ApplicationException. Exceptions of those classes generally mean that a non-recoverable problem has occurred.

Exception: On system-level or in a thread-routine, it is allowed to catch the Exception class directly, but only when approval by the Senior Designer has been obtained.

4.16 Bracing (C#)

Braces should never be considered optional. Even for single statement blocks, you should always use braces. This increases code readability and maintainability.

Example:

```
for (int i=0; i<100; i++) { DoSomething(i); }
```

Keep curly braces on lines by themselves. Don't put the opening curly brace at the end of the line. Open braces should always be at the beginning of the line after the statement that begins the block.

Curly braces should be indented to the same level as the code outside of the braces.

Contents of the brace should be indented by 4 spaces.

Example:

```
if (someExpression)
{
    DoSomething();
}
else
{
    DoSomethingElse();
}
```

Exception: For single line statements, such as the accessor "get" in a property that simply returns a variable: `get { return _privateVariable ; }`

"case" statements should be indented from the switch statement:

Example:

```
switch (someExpression)
{
    case 0:
        DoSomething();
        break;
    case 1:
        DoSomethingElse();
        break;
    case 2:
        {
            int n = 1;
            DoAnotherThing(n);
        }
        break;
}
```

5.0 Data-Specific Programming Guidelines

- Refer to the Information Resource Management: Database Development Standards Document.

5.1 Data Access

- If you are opening database connections, sockets, file stream etc, always close them in the "finally" block. This will ensure that even if an exception occurs after opening the connection, it will be safely closed in the "finally" block.
- Use try-catch in your data layer to catch all database exceptions. This exception handler should record all exceptions from the database. The details recorded should include the name of the command being executed, stored proc name, parameters, connection string used etc. After recording the exception, it could be re thrown so that another layer in the application can catch it and take appropriate action.

5.2 User Stored Procedures over inline SQL

Stored procedures give us many advantages over using SQL scripts and SQL statements that are dynamically created by our windows or web applications. Stored Procedures can be of great benefit in performance, security, readability, reusability, flexibility (conditional and procedural logic) and coding quality (restriction of coding to DBA / SQL specialist).

- Always use stored procedures over in-line SQL.

5.2.1 Rewriting inline SQL statements as Stored Procedures

Many applications make use of inline SQL statements that are created in the windows / web application and then passed to the database. An example of this would be a search box on an application where a user can enter a product name to search for that product. The application code that creates a dynamic SQL string might look something like:

```
' Build the SQL string by getting the product name out of a textbox named txtProductName
```

```
Dim SQL As String
```

```
SQL = "SELECT * FROM Products " & _
```

```
    " WHERE ProductName LIKE '%" & txtProductName.Text & "%' " & _
```

```
    " ORDER BY ProductName"
```

```
' Execute the dynamically generated SQL command
```

The above code could be replaced by coding the application to call a stored procedure in our database that did the product search. The stored procedure takes an input parameter of ProductName so it knows what products to search for:

```
CREATE PROCEDURE dbo.usp_ProductSearch
```

```
    @ProductName varchar(100)
```

```
AS
```

```
SET NOCOUNT ON
```

```
SELECT * FROM Products
```

```
WHERE ProductName LIKE '%' + @ProductName + '%'
```

```
ORDER BY ProductName
```

```
SET NOCOUNT OFF
```

```
GO
```

This procedure definition is longer than the dynamic SQL statement but when writing more complicated queries the stored procedure can actually be shorter. The above demonstrates the difference between some 'in-line' SQL that is generated in an application and a stored procedure database object that performs the same functionality.

5.2.2 Why use a Stored Procedure rather than in-line SQL

Compilation and storing of the query execution plan

When SQL Server runs a query it calculates the most efficient method of processing the query and stores it in something called the execution plan. Depending on the query, a good deal of the time taken to execute a query is actually spent on calculating the execution plan.

When inline SQL statements are executed the execution plan must be generated each time the query runs. When a stored procedure is called its execution plan is stored in memory and the next time the stored procedure is called the execution plan is retrieved from memory without needing to be recalculated. This increases the speed of execution and improves database performance.

Enabling of conditional and procedural logic

We cannot include any conditional logic when sending inline SQL statements from an application.

Stored Procedures enable us to tie as many SQL statements together with conditional logic such as IF ELSE, SELECT CASE and looping capabilities such as WHILE loops and CURSORS. This can lead to some very powerful querying techniques as well as dramatically reduce the number of times the application has to open and close a database connection or use the connection objects Execute command. This again helps improve database, network and therefore application performance.

Centralized repository for DML and DDL code enabling code reuse

Keeping all of our queries in Stored Procedures also means that the Data Manipulation Language (DML) and Data Definition Language (DDL) code is kept in one centralized place rather than scattered around various pages / forms / classes of one or more applications. This gives the following advantages:

SQL code is more likely to be written by a DBA or SQL developer rather than a web or non specialized SQL developer. This should result in better, more efficient code.

It isolates the code that is needed to reflect changes to the database structure thus making maintenance quicker

Procedures can be reused from different web pages / forms / classes

Procedures can be reused by more than one application

Code reuse means quicker development time

Code reuse means quicker debugging and maintenance time

Stored Procedures can be used to log errors in a standardized way by using the RAISEERROR function to write to the error log or by storing the error information in an 'Errors' table or emailing it to the DBA etc. This makes debugging the application easier by separating out data access errors from windows/web programming language errors (VB.NET / C# errors etc).

Protection from SQL Injection attacks

Inline SQL statements are at high risk from SQL injection attacks which can enable a hacker to compromise and wipe out your database.

Because Stored Procedures use parameters to receive user inputted values it prevents SQL injection from being used. There have been no known cases of SQL injection where static stored procedures are used.

Stored procedures parameters also define their data type and length e.g. a FirstName parameter might be of type varchar (25) so preventing any unsuitable variables of invalid data types or lengthier strings from being substituted into the query.

Enabling of strict security model

Frequently an application will define its connection string in one place and so all stored procedure calls will be made using these login credentials of the specified user. This means that this user's access to the database can be defined by only allowing them permissions to execute the allowed stored procedures whilst denying them permissions to the underlying database objects such as tables and views that are used by the stored procedures.

The benefit of this is that a hacker could not use the application connection to execute ad hoc SQL statements such as:

Perform a TRUNCATE TABLE

Execute a DELETE statement with no WHERE clause (deleting all rows from a table)

Perform an UPDATE statement with no WHERE clause.

SELECT sensitive data such as passwords

It also has the benefit that application users can only execute the stored procedures that implement the applications required business rules. It prevents non DBA's / SQL specialists writing poor ad hoc / inline SQL queries that don't conform to business rules. Any queries that do not follow the applications specified business rules can lead to invalid 'dirty' data in the database.

Readability

Static stored procedure code is a lot more readable than ad hoc dynamic SQL queries that are concatenated together in an application. The Stored Procedure editor window and Query Analyzer both include color formatting of SQL keywords as well as enabling the easy separation of different parts of a statement onto separate lines for easy reading.

5.3 Transactions

If you are performing multiple modifying statements - INSERT, UPDATE, or DELETES to one or more tables - and these actions, together, form a logical, atomic unit, then you'll likely want to utilize transactions.

A common example is when you have a parent/child relationship between two (or more) tables in the database. When deleting a row from the parent table you'll need to delete the associated child rows as well (or reassign them to a new parent).

The Common Steps of a Transaction

When using transactions you'll typically use the following sequence of steps:

1. Indicate that you want to start the transaction. All commands from this point forward are part of the logical, atomic operation.
2. Issue the discrete commands - the INSERT, UPDATE, and DELETES that make up your transaction.
3. If any of these commands cause an error, *rollback* the transaction. Rolling back a transaction has the effect of undoing the effects of all previous statements in the transaction.
4. If all steps succeed, *commit* the transaction. This persists the changes made throughout the transaction to the database.

Transactions are atomic so if there is any catastrophic failure - a loss of power, the database server crashing, etc. - when the database restarts the 'partially completed' transaction will be automatically rolled back, maintaining the consistency of the system.

When working with transactions through .NET you'll start the transaction and then issue a number of statements using that transaction object. A Try ... Catch block can be used to catch any exceptions raised by errors in issuing SQL statements that make up the transaction. In such a case you can rollback the transaction. If no errors occur, you can commit the transaction.

6.0 SQL Reporting Services

Know the 5 user experiences of Reporting Services: Vanilla, Website, Email, Windows, and SharePoint.

Know when to use Reporting Services.

Check that "RS Configuration Manager" is all green ticks.

Check out the built-in samples.

Know the 2 migration options to show your Access reports on the web.

Layout

Does your report print and display on the web correctly?

Include a useful footer at the bottom of your reports.

Avoid using word 'Report' in your reports.

Underline items with Hyperlink Action.

Show errors in Red.

Have a consistent report name.

Include feedback information in the report.

Know which version of Reporting Services you are running.

Data Layout

Show the past 6 months of totals in a chart.

Show data and chart in one.

Avoid using a single chart when you need it to be scaled.

Use expressions to show the correct scale on charts.

Show change in your reports.

Avoid showing change as a percentage.

Use alternating row colors.

Have nodes count like Outlook.

Avoid displaying decimal places

Have a consistent height of table row across all your reports.

Display zero number as blank in your reports.

Know the best way to show your data.

Show time format clearly.

Show all of the report parameters in body.

Know how to use logical page breaks.

Know how to prevent charts growing with rows.

Take advantage of vertical text (when lots of columns).

Data Logic

Use de-normalized database fields for calculated values.

Parameters

- Avoid showing empty reports by at least setting Default parameters.
- Avoid showing empty reports by the most intelligent default.
- Use the DateTime data type for date parameters.
- Have consistent parameter names.
- Cache popular reports for better performance.
- Schedule snapshots of slow reports for quicker access.

Internationalization

- Use regional friendly formatting.
- Make sure your language follows the user's regional settings.
- Be aware of your date format of parameter bar.
- Make sure your language rule has an exception for Currency Fields.

Admin

- Validate all your reports.
- Create a separate virtual directory for Admin access.
- Take advantage of 'Integrated Security' to do Payroll reports.
- Remove @ExecutionTime in subject of subscription email.
- Know to embed an RS report in ASP.NET page the right way (using Report Viewer instead of IFrame).
- Know how to get email list of report subscription.
- Add report owner in your report.
- Use a single line box instead of a double line box.

7.0 Project Settings, Structure, and Architecture

- Logically organize all your files within appropriate folders. Use 2 level folder hierarchies. You can have up to 10 folders in the root folder and each folder can have up to 5 sub folders. If you have too many folders than cannot be accommodated with the above mentioned 2 level hierarchy, you may need re factoring into multiple assemblies.
- Always use multi layer (N-Tier) architecture.
- Never access database from the UI pages. Always have a data layer class which performs all the database related tasks. This will help you support or migrate to another database back end easily.
- Separate your application into multiple assemblies. Group all independent utility classes into a separate class library. All your database related files can be in another class library.
- Do not use session variables throughout the code. Use session variables only within the classes and expose methods to access the value stored in the session variables. A class can access the session using `System.Web.HttpContext.Current.Session`
- Do not store large objects in session. Storing large objects in session may consume lot of server memory depending on the number of users.
- Always use style sheet to control the look and feel of the pages. Never specify font name and font size in any of the pages. Use appropriate style class. This will help you to change the UI of your application easily in future. Also, if you like to support customizing the UI for each customer, it is just a matter of developing another style sheet for them

7.1 Connection Strings

7.1.1 General Rules

Do not hard code connection strings within the forms or compiled code.

7.1.2 Web.Config

Declare your connection string under the <connectionStrings> section. This is automatically read and made available by ASP.NET when the application runs.

The connection string is not embedded in the Web.Config file using any encryption. Remember, ASP.NET is configured to ensure that a request for this file will not be handled. Meaning, if any one places a request for the file Web.Config, ASP.NET will not deliver it. (This is of course assuming that you have ensured that IIS is configured to disable directory file listing).

Regarding the User ID and Password in the connection string. First of all - if you are using SQL Server as the database, don't provide the User ID or Password. Instead opt for Windows Authentication. This is the best option. In such a case, you have two provisions. You can either add the web server's local ASPNET account to the DB Secured Logins and grant access to the DB or you can use the <identity impersonate="true" tag, and provide the windows UserID and Password in the Web.Config file.

8.0 Security

8.1 Input/Data Validation

If you make unfounded assumptions about the type, length, format, or range of input, your application is unlikely to be robust. Input validation can become a security issue if an attacker discovers that you have made unfounded assumptions. The attacker can then supply carefully crafted input that compromises your application. The misplaced trust of user input is one of the most common and serious vulnerabilities in Web applications.

To help avoid input data validation vulnerabilities:

- **Do not rely on ASP.NET request validation.**
- **Validate input for length, range, format, and type.**
- **Validate input from all sources like QueryString, cookies, and HTML controls.**
- **Do not rely on client-side validation.**
- **Avoid user-supplied file name and path input.**
- **Do not echo untrusted input.**
- **If you need to write out untrusted data, encode the output.**

8.1.1 Do Not Rely on ASP.NET Request Validation

The ASP.NET request validation feature performs basic input validation. Do not rely on it. Use it as an extra precautionary measure in addition to your own input validation. Only you can define what represents good input for your application.

Request validation is enabled by default. You can see this by examining the **validateRequest** attribute, which is set to **true** on the <pages> element in the Machine.config.comments file. Ensure that it is enabled for all pages except those that need to accept a range of HTML elements. If you need to disable it for a page, set the **ValidateRequest** attribute to **false** by using the @ **Page** directive.

8.1.2 Validate Input for Length, Range, Format, and Type

Do not trust input. An attacker passing malicious input can attempt SQL injection, cross-site scripting, and other injection attacks that aim to exploit your application's vulnerabilities. Check for known good data and constrain input by validating it for type, length, format, and range. For Web form applications that obtain input through server controls, use the ASP.NET validator controls, such as the **RegularExpressionValidator**, **RangeValidator**, and **CustomValidator**, to validate and constrain input. Check all numeric fields for type and range. If you are not using server controls, you can use regular expressions and the **Regex** class, and you can validate numeric ranges by converting the input value to an integer or double and then performing a range check.

8.1.3 Validate Input from All Sources Like QueryString, Cookies, and HTML Controls

Most Web applications accept input from various sources, including HTML controls, server controls, query strings, and cookies. Validate input from all of these sources to help prevent injection attacks. Use regular expressions to help validate input. The following example shows how to use the **Regex** class.

```
using System.Text.RegularExpressions ;
// Instance method:
Regex reg = new Regex(@"^[a-zA-Z'.\s]{1,40}$");
Response.Write(reg.IsMatch(Request.QueryString["Name"]));
// Static method:
if (!Regex.IsMatch(Request.QueryString["Name"], @"^[a-zA-Z'.\s]{1,40}$"))
{
    // Name does not match expression
}
```

If you cannot cache your regular expression for frequent use, you should use the static **IsMatch** method where possible for performance reasons, to avoid unnecessary object creation.

8.1.4 Do Not Rely on Client-Side Validation

Do not rely on client-side validation because it can be easily bypassed. For example, a malicious user could disable your client-side script routines by disabling JavaScript. Use client-side validation in addition to server-side validation to reduce round trips to the server and to improve the user experience.

8.1.5 Avoid User-Supplied File Name and Path Input

Where possible, avoid writing code that accepts user-supplied file or path input. Failure to do this can result in attackers coercing your application into accessing arbitrary files and resources. If your application must accept input file names, file paths, or URL paths, you need to validate that the path is in the correct format and that it points to a valid location within the context of your application.

8.1.5.1 File Names

Ensure that file paths only refer to files within your application's virtual directory hierarchy if that is appropriate. When checking file names, obtain the full name of the file by using the **System.IO.Path.GetFullPath** method.

8.1.5.2 File Paths

If you use **MapPath** to map a supplied virtual path to a physical path on the server, use the overloaded **Request.MapPath** method that accepts a **bool** parameter so that you can prevent cross-application mapping. The following code example shows this technique.

```
try
{
    string mappedPath = Request.MapPath( inputPath.Text, Request.ApplicationPath, false);
}
catch (HttpException)
{
    // Cross-application mapping attempted
}
```

The final **false** parameter prevents cross-application mapping. This means that a user cannot successfully supply a path that contains ".." to traverse outside of your application's virtual directory hierarchy. Any attempt to do this results in an exception of type **HttpException**.

8.1.6 Do Not Echo Untrusted Input

Do not echo input back to the user without first validating and/or encoding the data. Echoing input directly back to the user makes your application susceptible to malicious input attacks, such as cross-site scripting.

8.1.7 If You Need to Write Out Untrusted Data, Encode the Output

If you write output that includes user input or data from a shared database or a local file that you do not trust, encode it. Echoing input directly back to the user makes your application vulnerable to cross-site scripting attacks. Encoding the data ensures that it is treated as literal text and not as script. You can use the **HttpUtility.HtmlEncode** method. Similarly, if you write URLs that might contain unsafe characters because they have been constructed from input data or data from a shared database, use **HttpUtility.UriEncode** to make them safe.

Note: Make sure that you encode data at the last possible opportunity before the data is returned to the client.

8.1.8 Additional Resources

For additional resources on input validation, see the following How To's:

- [How To: Protect from Injection Attacks in ASP.NET](#)
- [How To: Use Regular Expressions to Constrain Input in ASP.NET](#)
- [How To: Protect from SQL Injection in ASP.NET](#)
- [How To: Prevent Cross-Site Scripting in ASP.NET](#)

8.2 Authentication

Where possible, you should use Windows authentication because this enables you to use an existing identity store such as your corporate Active Directory, it enables you to enforce strong password policies, you do not need to build custom identity store management tools and passwords are not transmitted over the network.

ASP.NET creates an **Identity** object that implements the **System.Security.Principal.IIdentity** interface to represent the authenticated user. Regardless of authentication method, you access the authenticated user through **HttpContext.Current.User**.

If your application is configured for Windows authentication, Internet Information Services (IIS) authenticates the user and the user's Windows token is passed to ASP.NET. ASP.NET constructs a **System.Security.Principal.WindowsIdentity** object, which contains the user's token, places this inside a **System.Security.Principal.WindowsPrincipal** object, and then attaches this to the current Web request.

When your application is configured for a non-Windows authentication mechanism such as forms authentication, the Windows token passed by IIS to ASP.NET is a token for the anonymous Internet user account IUSR_MACHINENAME. With forms authentication, ASP.NET constructs a **System.Web.Security.FormsIdentity** object and places it inside a **System.Security.Principal.GenericPrincipal** object before attaching it to the current Web request.

This section provides guidance on forms authentication and Windows authentication:

- **Forms authentication**
- **Windows authentication**

8.2.1 Forms Authentication

To protect forms authentication, you need to protect user-supplied credentials, the credential store and the authentication ticket. To do this, use the following guidelines:

- **Use membership providers instead of custom authentication.**
- **Use SSL to protect credentials and authentication cookies.**
- **If you cannot use SSL, consider reducing session lifetime.**
- **Validate user login information.**
- **Do not store passwords directly in the user store.**
- **Enforce strong passwords.**
- **Protect access to your credential store.**
- **Do not persist authentication cookies.**
- **Restrict authentication tickets to HTTPS connections.**
- **Consider partitioning your site to restricted areas and public areas.**
- **Use unique cookie names and paths.**

8.2.1.1 Use Membership Providers Instead of Custom Authentication

In ASP.NET version 1.1, you had to implement custom logic for validating user credentials, performing user management, and managing the authentication ticket. In version 2.0, you can use the built-in membership feature. The membership feature helps protect credentials, can enforce strong passwords, and provides consistent APIs for user validation and secure user management. The membership feature also automatically creates the authentication ticket for you.

The membership feature has built-in providers for user stores including SQL Server, Active Directory, and Active Directory Application Mode (ADAM). If you want to use an existing user store, such as a non-Active Directory LDAP directory, or a user store on another platform, create a custom membership provider inheriting from the **MembershipProvider** abstract base class. By doing this, your application can still benefit from using the standard membership features and API and login controls.

For more information, see [How To: Use Membership in ASP.NET 2.0](#).

8.2.1.2 Use SSL to Protect Credentials and Authentication Cookies

Use Secure Sockets Layer (SSL) to protect the authentication credentials and authentication cookies passed between browser and server. By using SSL, you prevent an attacker monitoring the network connection to obtain authentication credentials and capturing the authentication cookie to gain spoofed access to your application.

8.2.1.3 If You Cannot Use SSL, Consider Reducing Session Lifetime

If you cannot use SSL, limit the cookie lifetime to reduce the time window in which an attacker can use a captured cookie to gain access to your application with a spoofed identity. The default timeout for an authentication cookie is 30 minutes. Consider reducing this to 10 minutes as shown here.

```
<forms timeout="00:10:00" slidingExpiration="true" />
```

The `slidingExpiration="true"` setting ensures that the expiration period is reset after each Web request. With the preceding configuration, this means that the cookie only times out after a 10 minute period of inactivity.

If you are in a scenario where you are concerned about cookie hijacking, consider reducing the timeout and setting `slidingExpiration="false"`. If sliding expiration is turned off, the authentication cookie expires after the timeout period whether or not the user is active. After the timeout period, the user must re-authenticate.

8.2.1.4 Validate User Login Information

Validate user login information including user names and passwords for type, length, format, and range. Use regular expressions to constrain the input at the server. If you are not using the `SqlMembershipProvider` and need to develop your own queries to access your user store database, do not use login details to dynamically construct SQL statements because this makes your code susceptible to SQL injection. Instead, validate the input and then use parameterized stored procedures.

Also encode login details before echoing them back to the user's browser to prevent possible script injection. For example, use `HtmlEncode` as shown here.

```
Response.Write(HttpUtility.HtmlEncode("Welcome " + Request.Form["username"]));
```

8.2.1.5 Do Not Store Passwords Directly in the User Store

Do not store user passwords either in plaintext or encrypted format. Instead, store password hashes with salt. By storing your password with hashes and salt, you help prevent an attacker that gains access to your user store from obtaining the user passwords. If you use encryption, you have the added problem of securing the encryption key. Use one of the membership providers to help protect credentials in storage and where possible, specify a hashed password format on your provider configuration.

If you must implement your own user stores, store one-way password hashes with salt. Generate the hash from a combination of the password and a random salt value. Use an algorithm such as SHA256. If your credential store is compromised, the salt value helps to slow an attacker who is attempting to perform a dictionary attack. This gives you additional time to detect and react to the compromise.

8.2.1.6 Enforce Strong Passwords

Ensure that your passwords are complex enough to prevent users guessing other users' passwords and to prevent successful dictionary attacks against your user credential store.

By default, the ASP.NET membership providers enforce strong passwords. For example, the **SqlMembershipProvider** and the **ActiveDirectoryMembership** providers ensure that passwords are at least seven characters in length with at least one non-alphanumeric character. Ensure that your membership provider configuration enforces passwords of at least this strength. To configure the precise password complexity rules enforced by your provider, you can set the following additional attributes:

- **passwordStrengthRegularExpression**. The default is "".
- **minRequiredPasswordLength**. The default is 7.
- **minRequiredNonalphanumericCharacters**. The default is 1.

Note: The default values shown here apply to the **SqlMembershipProvider** and the **ActiveDirectoryMembershipProvider**. The **ActiveDirectoryMembershipProvider** also verifies passwords against the default domain password policy.

8.2.1.7 Protect Access to Your Credential Store

Ensure only those accounts that require access are granted access to your credential store. This helps to protect the credential store by limiting access to it. For example, consider limiting access to only your application's account. Ensure that the connection string used to identify your credential store is encrypted.

Also consider storing your credential database on a physically separate server from your Web server. This makes it harder for an attacker to compromise your credential store even if he or she manages to take control of your Web server.

8.2.1.8 Do Not Persist Authentication Cookies

Do not persist authentication cookies because they are stored in the user's profile and can be stolen if an attacker gets physical access to the user's computer. To ensure a non-persistent cookie, set the **DisplayRememberMe** property of the **Login** control to **false**. If you are not using the login controls, you can specify a non-persistent cookie when you call either the **RedirectFromLoginPage** or **SetAuthCookie** methods of the **FormsAuthentication** class having validated the user's credentials, as shown here.

```
public void Login_Click(object sender, EventArgs e)
{
    // Is the user valid?
    if (Membership.ValidateUser(userName.Text, password.Text))
    {
        // Parameter two set to false indicates non-persistent cookie
        FormsAuthentication.RedirectFromLoginPage(username.Text, false);
    }
    else
    {
        Status.Text = "Invalid credentials. Please try again.";
    }
}
```

8.2.1.9 Restrict Authentication Tickets to HTTPS Connections

Set the **secure** property of the authentication cookie to ensure that browsers only send authentication cookies over HTTPS connections. By using SSL, you prevent an attacker from capturing the authentication cookie to gain spoofed access to your application.

Set the **secure** property by using **requireSSL="true"** on the **<forms>** element as shown here.

```
<forms loginUrl="Secure\Login.aspx" requireSSL="true" />
```

8.2.1.10 Consider Partitioning Your Site to Restricted Areas and Public Areas

To avoid the performance overhead of using SSL across your entire site, consider using a separate folder to help protect pages that require authenticated access. Configure that folder in IIS to require SSL access. Those pages that support anonymous access can safely be accessed over HTTP connections.

8.2.1.11 Use Unique Cookie Names and Paths

Use unique name and path attribute values on the **<forms>** element. By ensuring unique names, you prevent possible problems that can occur when hosting multiple applications on the same server. For example, if you do not use distinct names, it is possible for a user who is authenticated in one application to make a request to another application without being redirected to that application's logon page.

8.2.1.12 Additional Considerations

In addition to the preceding guidelines, consider the following.

- **Set httpOnly on the authentication cookie.** Set the **httpOnlyCookies** attribute on the authentication cookie. Internet Explorer 6 Service Pack 1 supports this attribute, which prevents client-side script from accessing the cookie from the **document.cookie** property. The **System.Net.Cookie** class in .NET Framework version 2.0 supports an **HttpOnly** property. The **HttpOnly** property is always set to true by forms authentication.

8.2.1.13 Additional Resources

For additional resources on forms authentication, see the following How To's:

- [How To: Protect Forms Authentication in ASP.NET 2.0](#)
- [How To: Use Forms Authentication with SQL Server in ASP.NET 2.0](#)
- [How To: Use Forms Authentication with Active Directory in ASP.NET 2.0](#)
- [How To: Use Forms Authentication with Active Directory in Multiple Domains in ASP.NET 2.0](#)

8.2.2 Windows Authentication

If you use Windows authentication, consider the following guidelines:

- **Choose Windows authentication when you can.**
- **Enforce strong password policies.**

8.2.2.1 Choose Windows Authentication When You Can

When possible, use Windows authentication to authenticate your users. By using Windows authentication with Active Directory, you benefit from a unified identity store, centralized account administration, enforceable account and password policies, and strong authentication that avoids sending passwords over the network.

8.2.2.2 Enforce Strong Password Policies

To help ensure that users cannot guess one another's passwords and to help prevent successful dictionary attacks in the event your password store is compromised, enforce strong passwords through Active Directory policy. To enforce a strong password policy:

- **Set password length and complexity.** Strong passwords are eight or more characters and must include both alphabetical and numeric characters. The default enforced by the **ActiveDirectoryMembershipProvider** is seven characters with at least one non-alphanumeric character. If you use this provider, the provider settings are checked first, followed by the Active Directory settings. Users must supply passwords that conform to the stronger of the two.
- **Set password expiration.** Passwords that expire regularly reduce the likelihood that an old password can be used for unauthorized access. Frequency of expiration is usually guided by a company's security policy. You should define this in Active Directory.

For more information on forms authentication, see [How To: Use Windows Authentication in ASP.NET 2.0](#).

8.3 Authorization

You control authorization administratively through URL and file authorization and programmatically by performing identity and role checks in code.

You also need to consider who you are authorizing. Are you authorizing the application, the caller, or both? Also, what are you authorizing access to? You could be authorizing against:

- **System resources.** These include the file system and registry.
- **Application resources.** These include business logic and network resources, such as databases and Web services.
- **User resources.** These include resources such as customer records.

When implementing authorization logic, consider the following guidelines:

- **Use URL authorization for page and directory access control.**
- **Configure ACLs on your Web site files.**
- **Use ASP.NET role manager for roles authorization.**
- **If your role lookup is expensive, consider role caching.**
- **Protect your authorization cookie.**

8.3.1 Use URL Authorization for Page and Directory Access Control

Use URL authorization to control which users and groups of users have access to the application or to parts of the application. In ASP.NET version 1.1, URL authorization only applies to file types that are mapped by IIS to the ASP.NET ISAPI extension (Aspnet_isapi.dll). In ASP.NET version 2.0 on Windows Server 2003, URL authorization protects all files in a directory, even files that are not mapped to ASP.NET, such as .html, .gif, and .jpg files.

When your application uses Windows authentication, you can authorize access to Windows user and/or Windows group accounts. To do so, you should configure your application to use ASP.NET Role Manager. For more information, see the "Use ASP.NET Role Manager for Roles Authorization" section in this topic.

You can use the `<location>` tag to apply authorization settings to an individual file or directory. The following example shows how you can apply authorization to a specific file (page.aspx).

```
<location path="page.aspx" />
  <authorization>
    <allow users="DomainName\Bob, DomainName\Mary" />
    <deny users="*" />
  </authorization>
</location>
```

8.3.2 Configure ACLs on Your Web Site Files

You need to configure the right access control lists (ACLs) for the right identities on your Web site files so that IIS and also ASP.NET file authorization control access to these files appropriately. You need to grant access to the following identities:

- **Your Web application identity.** If you are using a custom service account to run your ASP.NET application, you can grant the appropriate permissions to the IIS metabase and to the file system by running `Aspnet_regiis.exe` with the `-ga` switch.
- **Your application's users.** ASP.NET file authorization performs access checks for file types mapped by IIS to the ASP.NET ISAPI extension (`Aspnet_isapi.dll`). If you are using Windows authentication, the authenticated user's Windows access token (which may be `IUSR_MACHINE` for anonymous users) is checked against the ACL attached to the requested ASP.NET file. If you are using forms authentication, access is checked against `IUSR_MACHINE`.

File authorization works automatically when using Windows authentication, and there is no need to impersonate the original user. The **FileAuthorizationModule** only performs access checks against the requested file. For example, if you request `Default.aspx` and it contains an embedded user control (`Usercontrol.ascx`), which in turn includes an image tag (pointing to `Image.gif`), the **FileAuthorizationModule** performs an access check for `Default.aspx` and `Usercontrol.ascx`, because these file types are mapped by IIS to the ASP.NET ISAPI extension. The **FileAuthorizationModule** does not perform a check for `Image.gif`, because this is a static file handled internally by IIS. However, because access checks for static files are performed by IIS, the authenticated user must still be granted read permission to the file with an appropriately configured ACL.

8.3.3 Use ASP.NET Role Manager for Roles Authorization

In ASP.NET version 1.1, you had to create, manage, and look up roles for the authenticated user by writing your own code. ASP.NET version 2.0 provides a new role manager feature that automatically performs this work. Roles are accessed from the configured role store by the **RoleManager** HTTP module by using the configured role provider. This occurs after the user is authenticated but before URL authorization and file authorization access checks occur and before programmatic role checks can occur.

If your application needs role-based authorization, use the following guidelines:

- **Use role providers to perform role authorization.** Role providers allow you to load the roles for users without writing and maintaining code. Additionally, the role providers offer a consistent way for you to check the role membership of your users, regardless of the underlying data store. Where possible, use one of the supplied roles providers; these include **SqlRoleProvider**, **WindowsTokenRoleProvider**, and **AuthorizationStoreRoleProvider**. If you already have a custom role store in a non-SQL Server database or in a non-Active Directory LDAP store, consider implementing your own custom role provider.

The following code shows how to use the role manager API and specifically **Roles.IsUserInRole**.

```
// Tests whether the currently authenticated user is a member
// of a particular role.
if(Roles.IsUserInRole("Manager"))
    // Perform restricted operation
else
    // Return unauthorized access error.
// Tests whether any given user is a member of a role:
if(Roles.IsUserInRole("Bob","Manager"))
    // Perform restricted operation
else
    // Return unauthorized access error.
```

- **Use the `SqlRoleProvider` when your roles are in SQL Server.** If you store role information in SQL Server, configure your application to use the `SqlRoleProvider`. You can also configure forms authentication with the `SqlMembershipProvider` to use the same database for authentication and authorization, although this is not required.
- **Use the `WindowsTokenRoleProvider` when you use Windows groups as roles.** If your application uses Windows authentication and you use Windows groups as roles, configure your application to use the `WindowsTokenRoleProvider`.
- **Use the `AuthorizationStoreRoleProvider` when your application roles are in ADAM.** If your application uses Windows authentication against Active Directory, and you need application specific roles instead of using your domain Windows group membership, you can store role information in SQL Server or in an Authorization Manager (AzMan) policy store in ADAM. Authorization Manager provides a Microsoft Management Console (MMC) snap-in, to create and manage roles, and to manage role membership for users.

If your user accounts are in Active Directory, but you cannot use Windows authentication and must use forms authentication, a good solution for roles management is to use the `AuthorizationStoreRoleProvider` with an AzMan policy store in ADAM.

Note: The `AuthorizationStoreRoleProvider` does not directly support Authorization Manager business logic such as operations and tasks. To do this, you must use `P/Invoke` and call the Authorization Manager API directly.

For more information, see [How To: Use Role Manager in ASP.NET 2.0](#).

8.3.4 If Your Role Lookup Is Expensive, Consider Role Caching

If the performance overhead of role lookup is too great, perhaps because of slow data access or a large number of roles, consider caching roles in the roles cookie by setting the `cacheRolesInCookie` attribute to `true` in the `Web.config` file as shown here.

```
<roleManager enabled="true" cacheRolesInCookie="true">
</roleManager>
```

When role checks are performed, the roles cookie is checked before calling the role provider to check the list of roles within the data source. This improves performance. The cookie is dynamically updated to cache the most recently validated role names. If the role information for a user is too long to store in a cookie, ASP.NET stores only the most recently used role information in the cookie and then it looks up additional role information in the data source as required. The most recently referred to roles end up being cached in the cookie.

8.3.5 Protect Your Authorization Cookie

You should protect roles data in the authorization cookie to stop users from modifying the list of roles to which they belong, and to stop intruders from gaining information about the roles used by your application. You protect the authorization cookie by appropriately configuring the `<roleManager>` element as follows:

- Set the `cookieProtection` attribute to `All`. This ensures that the role names in the cookie are digitally signed and encrypted to prevent unauthorized viewing and modification of the role data.
- Set the `cookieSlidingExpiration` attribute to `true` and the `cookieTimeout` attribute to an integer number of minutes (for example, 10 to 20 or fewer). The cookie expires when the timeout expires and must be renewed.
- Set the `cookieRequireSSL` attribute to `true` to specify that the authorization cookie with the role information should only be returned to the server over HTTPS connections.
- Set the `createPersistentCookie` attribute to `false` to ensure that the roles cookie is session-based and not a persistent cookie.
- If you cannot use SSL, consider reducing the `cookieTimeout` to 10 minutes. If you are concerned with cookie hijacking, consider setting the `cookieSlidingExpiration` attribute to `false`. This causes the cookie to expire after the fixed timeout period and must be renewed.

The following example shows a `<roleManager>` element configured to protect the authorization cookie.

```
<roleManager enabled="true"
  cacheRolesInCookie="true"
  cookieName=".ASPROLES"
  cookieTimeout="30"
  cookiePath="/"
  cookieRequireSSL="true"
  cookieSlidingExpiration="true"
  cookieProtection="All"
  createPersistentCookie="false">
</roleManager>
```

8.3.6 Additional Resources

For additional resources on Windows authentication, see the following How To's:

- [How To: Use ADAM for Roles in ASP.NET 2.0](#)
- [How To: Use Authorization Manager \(AzMan\) with ASP.NET 2.0](#)
- [How To: Use the Network Service Account to Access Resources in ASP.NET](#)

8.4 Code Access Security

Code access security is a resource constraint model designed to restrict the types of system resource that code can access and the types of privileged operation that the code can perform. These restrictions are independent of the user who calls the code or the user account under which the code runs. ASP.NET code access security policy is defined by trust levels.

When using code access security with ASP.NET, consider the following guidelines:

- **Consider code access security for partial trust applications.**
- **Choose a trust level that does not exceed your application's requirements.**
- **Create a custom trust policy if your application needs additional permissions.**
- **Use Medium trust in shared hosting environments.**

8.4.1 Consider Code Access Security for Partial Trust Applications

If your application calls only managed code, you can use different code access security trust levels to incrementally limit your exposure to security attacks and provide extra degrees of application isolation. You should consider code access security particularly if you need application isolation in shared hosting environments.

If you do not plan on running your application at a partial trust level, code access security presents no additional design or development considerations because at Full trust, code access permission demands will succeed. An application's trust level is determined by the `<trust>` element as shown here.

```
<trust level="Full|High|Medium|Low|Minimal" />
```

8.4.2 Choose a Trust Level That Does Not Exceed Your Application's Requirements

Choose a trust level that does not provide additional permissions beyond what your application requires. By doing this, you follow the principle of least privilege and constrain your application as much as possible. To select the most appropriate trust level, identify the precise set of code access security permissions that your application requires. You can do this by manually reviewing your code or by using the Permissions Calculator tool (Permcals.exe). Evaluate whether the permissions required for your application match those provided by any of the standard trust levels. To see the permissions each trust level provides, examine each trust level policy file in the %windir%\Microsoft.NET\Framework\{version}\CONFIG directory, beginning with the High trust policy file web_hightrust.config.

If your application requires fewer code access security permissions than those provided by the High trust level, move on to consider Medium trust. Repeat the process, moving from Medium to Low to Minimal, and keep evaluating the partial trust levels until you reach an exact match to your application's requirements or until your application's required permissions slightly exceed a partial trust level. If your application needs more permissions than are granted by one level but requires fewer than are provided by the next level, consider creating a custom trust policy.

For more information, see [How To: Use Code Access Security in ASP.NET 2.0](#).

8.4.3 Create a Custom Trust Policy if Your Application Needs Additional Permissions

If your application requires additional permissions beyond those provided at a particular trust level, but it does not need the additional permissions provided by the next trust level, create a custom trust policy file. This avoids granting your application unnecessary permissions.

To create a custom policy, copy the trust level policy file that most closely matches your application's permission requirements to create a new custom policy file. Locate this file in the standard policy file directory, which is %Windir%\Microsoft.NET\Framework\{Version}\CONFIG. Then add the required additional permissions to the custom policy file and configure your application to run using the custom policy.

For more information, see [How To: Use Code Access Security in ASP.NET 2.0](#).

8.4.4 Use Medium Trust in Shared Hosting Environments

In ASP.NET version 1.1, Web applications configured for Medium trust could not access SQL Server databases. In ASP.NET version 2.0, SQL Server database access is available to Medium trust applications because the SQL Server managed data provider no longer demands Full trust. If you need to isolate multiple applications on a shared server from one another and from shared system resources, use Medium trust. To enforce Medium trust for all Web applications on a server, use the following configuration in the machine-level Web.config file.

```
<location allowOverride="false">
  <system.web>
    <trust level="Medium" originUrl="" />
  </system.web>
</location>
```

By setting **allowOverride="false"**, an individual developer is unable to override the Medium trust policy setting in his or her application's Web.config file.

Medium trust provides application isolation because it restricts file system access to the application's own virtual directory hierarchy, and it limits access to HTTP Web resources to a defined address or set of addresses specified by the originUrl attribute on the <trust> element. It also prevents access to shared system resources such as the Windows event log and registry. Additionally, Medium trust applications cannot use reflection and cannot access non-SQL Server OLE DB data sources.

For more information, see [How To: Use Medium Trust in ASP.NET 2.0](#).

8.5 Data Access

When building the data access layer of your application, consider the following guidelines:

- **Encrypt your connection strings.**
- **Use least-privileged accounts for database access.**
- **Use Windows authentication where possible.**
- **If you use Windows authentication, use a trusted service account.**
- **If you cannot use a domain account, consider mirrored accounts.**
- **When using SQL authentication, use strong passwords.**
- **When using SQL authentication, protect credentials over the network.**
- **When using SQL authentication, protect credentials in configuration files.**
- **Validate untrusted input passed to your data access methods.**
- **When constructing SQL queries, use type safe SQL parameters.**
- **Avoid dynamic queries that accept user input.**

8.5.1 Encrypt your connection strings

In ASP.NET version 1.1, you could encrypt connection strings by using DPAPI encryption, although you had to write managed code to wrap the calls to DPAPI. This approach also presented problems in Web farms because of machine affinity. In ASP.NET version 2.0, you can use a protected configuration provider, such as DPAPI or RSA, which is easier to use in Web farms. You do not have to write code because you use the `Aspnet_regiis.exe` utility.

In ASP.NET version 2.0 applications, place your database connection strings inside the `<connectionStrings>` element of the `Web.config` file and then encrypt that element by using the `Aspnet_regiis` utility. In a Web farm, use the RSA-protected configuration provider because RSA keys can be exported and imported across servers.

For more information, see [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](#) and [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](#).

8.5.2 Use Least-Privileged Accounts for Database Access

Your application should connect to the database by using a least-privileged account. This limits the damage that can be done in the event of a SQL injection attack or in the event of an attacker obtaining your account's credentials. With Windows authentication, use a least-privileged account with limited operating system permissions, and limited ability to access Windows resources. Regardless of authentication mechanism, restrict the account's permissions in the database.

Use the following pattern to limit permissions in the database:

1. Create a SQL Server login for the account.
2. Map the login to a database user in the required database.
3. Place the database user in a database role.
4. Grant the database role limited permissions to only those stored procedures or tables that your application needs to access.

Ideally, provide no direct table access and limit access to selected stored procedures only. If you must grant table access, grant the minimum access that the application requires. For example, do not grant update access if read access is sufficient.

By using a database role, you avoid granting permissions directly to the database user. This isolates you from potential changes to the database user name. For example, if you change the database user name, you can simply add the new user to the database role and remove the existing one.

8.5.3 Use Windows Authentication Where Possible

Prefer Windows authentication when connecting to SQL Server or other databases that support it. Windows authentication offers a number of security advantages in comparison to SQL authentication:

- Accounts are centralized and managed by your Active Directory or local authority store.
- Strong password policies can be controlled and enforced by your domain or local security policy.
- Passwords are not transmitted over the network.
- User IDs and passwords are not specified in database connection strings.

For more information, see [How To: Connect to SQL Server Using Windows Authentication in ASP.NET 2.0](#).

8.5.4 If You Use Windows Authentication, Use a Trusted Service Account

If you use Windows authentication, use a trusted service account to access the database when possible. This is usually your application's process account. By using a single trusted service account, your application benefits from connection pooling; this provides greater scalability. Also, account administration and authorization within the database is simplified. If you need per-user authorization in the database or need to use operating system auditing to track the activity of individual users, you need to use impersonation and delegation and access the database using the caller's identity. This approach has limited scalability because it prevents the efficient use of connection pooling.

8.5.5 If You Cannot Use a Domain Account, Consider Mirrored Accounts

If you cannot use domain accounts because of domain trust or firewall restrictions, consider using mirrored service accounts instead. With this approach, you still use Windows authentication, but you create two local accounts with the same name and password on the Web server and database server. You configure your application to run using the local account created on the Web server and create a SQL login for the local account on the database server. With this approach, you must ensure that the passwords of the two accounts remain in synchronization.

For more information, see [How To: Connect to SQL Server Using SQL Authentication in ASP.NET 2.0](#).

8.5.6 When Using SQL Authentication, Use Strong Passwords

If you use SQL Server authentication, make sure you use a least-privileged account with a strong password to prevent an attacker guessing your account's password. A strong password should be at least seven characters in length and contain a combination of alphabetic, numeric, and special characters.

Avoid using blank passwords and the sa account as shown in the following connection string.

```
SqlConnectionString = "Server=YourServer\Instance; Database=YourDatabase; uid=sa; pwd=;"
```

Use least-privileged accounts with a strong password, such as the following.

```
SqlConnectionString= "Server=YourServer\Instance;  
    Database=YourDatabase;  
    uid=YourStrongAccount;  
    pwd=YourStrongP@ssw0rd;"
```

8.5.7 When Using SQL Authentication, Protect Credentials Over the Network

If your application is not located in a physically secure isolated data center, you should use Internet Protocol Security (IPSec) or SSL to create an encrypted communication channel between the Web server and database server if you use SQL authentication. Failure to do this means that credentials can be easily captured with a network monitor. When you connect to SQL Server with SQL authentication, the credentials are not encrypted prior to transmission across the network.

Use SSL when you need granular channel protection for a particular application instead of for all applications and services running on a computer. If you want to protect all the IP traffic between Web and database server, use IPSec. You can also use IPSec to restrict which computers can communicate with one another. For example, you can help protect a database server by establishing a policy that permits requests only from a trusted client computer, such as an application or Web server. You can also restrict communication to specific IP protocols and TCP/UDP ports.

8.5.8 When Using SQL Authentication, Protect Credentials in Configuration Files

To protect credentials in configuration files, encrypt them. Place your database connection strings inside the `<connectionStrings>` element of the Web.config file and then encrypt that element by using the Aspnet_regiis utility. You can use DPAPI or RSA encryption. Use RSA in Web farms because you can easily export and import RSA keys across servers. Protecting connection strings is particularly important for connection strings that use SQL authentication because they contain clear text user IDs and passwords.

Note: You should also encrypt connection strings if you use Windows authentication. Although this form of connection string does not contain credentials, you should aim to keep server and database names private.

For more information, see [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](#) and [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](#).

8.5.9 Validate Untrusted Input Passed to Your Data Access Methods

If your data access methods receive input parameters from outside the trust boundary of your data access code, make sure you validate them for type, length, format, and range. You can use regular expressions for text input and perform type and range checks on numeric data. If you do not do this, your data access code is potentially susceptible to SQL injection.

Only omit input parameter validation in your data access methods if you know for certain that data can only be supplied by trusted code, such as your application's business logic, which you know has thoroughly validated the data passed to it.

Note: Avoid storing encoded data; instead, encode the data as close as possible to the output.

8.5.10 When Constructing SQL Queries, Use Type Safe SQL Parameters

Use type safe parameters when constructing SQL queries to avoid possible SQL injection attacks that can occur with unfiltered input. You can use type safe parameters with stored procedures and with dynamic SQL statements. Parameters are treated as literal values by the database and not as executable code. Parameters are also checked for type and length.

The following code shows how to use type safe parameters with the **SqlParameterCollection** when calling a stored procedure.

```
using System.Data;
using System.Data.SqlClient;
using (SqlConnection connection = new SqlConnection(connectionString))
{
    DataSet userDataset = new DataSet();
    SqlDataAdapter myCommand = new SqlDataAdapter(LoginStoredProcedure", connection);
    myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
    myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11);
    myCommand.SelectCommand.Parameters["@au_id"].Value = SSN.Text;
    myCommand.Fill(userDataset);
}
```

In the preceding code example, the input value cannot be longer than 11 characters. If the data does not conform to the type or length defined by the parameter, the **SqlParameter** class throws an exception.

For more information about preventing SQL injection, see [How To: Protect from SQL Injection in ASP.NET](#).

8.5.11 Avoid Dynamic Queries That Accept User Input

Avoid constructing SQL queries in code that include user input; instead, prefer parameterized store procedures that use type safe SQL parameters. If you construct queries dynamically using user input, your code is susceptible to SQL injection. For example, avoid the following style of code.

```
// Use dynamic SQL
SqlDataAdapter myCommand = new SqlDataAdapter(
    "SELECT au_lname, au_fname FROM authors WHERE au_id = " +
    SSN.Text + "'", myConnection);
```

If a malicious user supplies " ; DROP DATABASE pubs --" for the SSN.Text field, the code inserts the user's malicious input and generates the following query.

```
SELECT au_lname, au_fname FROM authors WHERE au_id = "; DROP DATABASE pubs --'
```

The ; (semicolon) character tells SQL that this is the end of the current statement, which is then followed by the following malicious SQL code, which in this example drops the authors table.

8.5.12 Additional Resources

For additional resources on connecting to SQL Server, see the following How To's:

- [How To: Connect to SQL Server Using Windows Authentication in ASP.NET 2.0](#)
- [How To: Connect to SQL Server Using SQL Authentication in ASP.NET 2.0](#)

8.6 Exception Management

Correct exception handling in your Web pages prevents sensitive exception details from being revealed to the user, improves application robustness, and helps avoid leaving your application in an inconsistent state in the event of errors. Consider the following guidelines:

- **Use structured exception handling.**
- **Do not reveal exception details to the client.**
- **Use a global error handler to catch unhandled exceptions.**

8.6.1 Use Structured Exception Handling

Use structured exception handling and catch exception conditions. Doing this improves robustness and avoids leaving your application in an inconsistent state that may lead to information disclosure. It also helps protect your application from denial of service attacks. Use **finally** blocks to ensure that resources are cleaned up and closed even in the event of an exception condition. Decide how to propagate exceptions internally in your application and give special consideration to what occurs at the application boundary.

8.6.2 Do Not Reveal Exception Details to the Client

When exceptions occur, return concise error messages to the client and log specific details on the server. Do not reveal internal system or application details, such as stack traces, SQL statement fragments, and table or database names to the client. Ensure that this type of information is not allowed to propagate to the end user or beyond your current trust boundary. A malicious user could use system-level diagnostic information to learn about your application and probe for weaknesses to exploit in future attacks.

Prevent detailed error messages from displaying by setting the **mode** attribute of the `<customErrors>` element in the Web.config file to **On**, so that all callers receive filtered exception information. Do not use **mode="Off"** because this allows detailed error pages intended for application developers that contain system-level information to be returned to the client.

You should also use the `<customErrors>` section of the Web.config file as shown in the following code example to specify a default error page to display, along with other required error pages for specific HTTP response codes that indicate errors.

```
<customErrors mode="On" defaultRedirect="ErrDefault.aspx">
  <error statusCode="401" redirect="ErrUnauthorized.aspx" />
  <error statusCode="404" redirect="ErrPageNotFound.aspx" />
  <error statusCode="500" redirect="ErrServer.htm" />
</customErrors>
```

Example: A status code of 404 tells ASP.NET to display custom errors from a remote client or a local client and to display a page named errorPage.aspx. Error "404" is "Page not found" error.

The **defaultRedirect** attribute allows you to use a custom error page for your application, which, for example, might include support contact details. Use these application-wide error pages to provide user-friendly responses for errors that are not caught in a structured event handling.

If custom error mode is turned "OFF" than you will see ASP.NET default error message. This error messages are good for debugging purposes but should never be exposed to the users. The users should always be presented with friendly errors if any.

8.6.3 Use a Global Error Handler to Catch Unhandled Exceptions

Define an application-level global error handler in Global.asax to catch any exceptions that are not handled in code. Do this to avoid accidentally returning detailed error information to the client. You should log all exceptions in the event log to record them for tracking and later analysis. Use code similar to the following.

```
<%@ Application Language="C#" %>
<%@ Import Namespace="System.Diagnostics" %>
<script language="C#" runat="server">
void Application_Error(object sender, EventArgs e)
{
    //get reference to the source of the exception chain
    Exception ex = Server.GetLastError().GetBaseException();
    // log the details of the exception and page state to the
    // event log.
    EventLog.WriteEntry("My Web Application",
        "MESSAGE: " + ex.Message +
        "\nSOURCE: " + ex.Source,
        EventLogEntryType.Error);
    // Optional e-mail or other notification here...
}
</script>
```

8.7 Impersonation/Delegation

By default, ASP.NET Web applications do not impersonate, although in some scenarios, you need to use impersonation to perform operations or access resources using the authenticated user's identity. If you use impersonation, consider the following guidelines:

- **Know your tradeoffs with impersonation.**
- **Avoid calling LogonUser.**
- **Avoid programmatic impersonation where possible.**
- **If you need to impersonate, consider threading issues.**
- **If you need to impersonate, clean up appropriately.**
- **Avoid losing impersonation tokens.**

8.7.1 Know Your Tradeoffs with Impersonation

Be aware that impersonation prevents the efficient use of connection pooling if you access downstream databases by using the impersonated identity. This impacts the ability of your application to scale. Also, using impersonation can introduce other security vulnerabilities, particularly in multi-threaded applications, such as ASP.NET Web applications.

You might need impersonation if you need to:

- Flow the original caller's security context to the middle tier and/or data tier of your Web application to support fine-grained (per-user) authorization.
- Flow the original caller's security context to the downstream tiers to support operating system level auditing.
- Access a particular network resource by using a specific identity.

8.7.2 Avoid Calling LogonUser

Avoid writing code that calls the **LogonUser** API to create impersonation tokens because this forces you to store user names and passwords on your Web server.

Instead, use a unique application pool with a specific process identity if you need a specific identity to access downstream resources. If you need multiple identities to access a range of downstream resources and services, use Windows Server 2003 protocol transition and the new **WindowsIdentity** constructor; this allows you to create a Windows token that is given only an account's user principal name (UPN). To access a network resource, you need to delegate-level token. To get this token type, your server needs to be configured as trusted for delegation in Active Directory.

The following code shows how to use this constructor to obtain a Windows token for a given user.

```
using System;
using System.Security.Principal;
public void ConstructToken(string upn, out WindowsPrincipal p)
{
    WindowsIdentity id = new WindowsIdentity(upn);
    p = new WindowsPrincipal(id);
}
```

Note: An account's UPN is guaranteed to be unique within a forest. Frequently, the UPN is the user's e-mail address, but it does not have to be. A user always has a UPN and by default, it is *userlogonname@fullyqualifieddomainname*. If you are logged into a domain, you can find your UPN name by running **whoami /upn** from a command window.

For more information, see [How To: Use Protocol Transition and Constrained Delegation with ASP.NET 2.0](#).

8.7.3 Avoid Programmatic Impersonation Where Possible

If programmatic impersonation is not done properly, it can introduce security vulnerabilities. It is difficult to get it correct, particularly in multithreaded applications. When possible, use alternative approaches, such as a custom domain process identity for resource access. You should avoid programmatic impersonation where possible for the following reasons:

- It is easy to introduce errors because of thread switches where the thread impersonation token is not propagated across threads.
- Some programmatic techniques require you to store credentials which should be avoided.
- Some programmatic techniques require you to grant additional privileges to your process account, which you should avoid. For example, you must grant your process account "Act as part of the operating system" if you call **LogonUser** on Windows Server 2000 or to obtain an impersonate-level token on Windows Server 2003 when you use the new **WindowsIdentity** constructor that generates a token from a user principal name.
- If exceptions occur while impersonating, it is possible for malicious code higher in the call stack to run using the impersonated identity. This can present security issues, particularly if you impersonate a highly privileged account.

8.7.4 If You Need to Impersonate, Consider Threading Issues

Losing an impersonated security context because of thread switches is a common vulnerability. The common language runtime (CLR) automatically propagates impersonation tokens to threads that you create using any of the managed threading techniques, such as **Thread.Start**, an asynchronous delegate, or **QueueUserWorkItem**. However, it is easy to drop the thread impersonation token if you use COM interop with components that have incompatible threading models or if you use unmanaged techniques to create new threads such as the Win32 **CreateThread** API.

8.7.5 If You Need to Impersonate, Clean Up Appropriately

If you must use programmatic impersonation, use structured exception handling and put the impersonation code inside **try** blocks. Use a **catch** block to handle exceptions and use a **finally** block to ensure that the impersonation is reverted as shown here.

```
using System.Security.Principal;
WindowsIdentity winIdentity = new WindowsIdentity("username@domainName");
WindowsImpersonationContext ctx = winIdentity.Impersonate();
try
{
    // Do work
}
catch(Exception ex)
{
    // Stop impersonating
    ctx.Undo();
}
finally
{
    // Stop impersonating
    ctx.Undo();
}
```

By using a **finally** block, you ensure that the impersonation token is removed from the current thread whether an exception is generated or not. Also be aware that if your code fails to catch exceptions, a malicious user could use exception filters to execute code that runs under the impersonated security context. This is particularly serious if your code impersonates a privileged account. If your code does not catch the exception, exception filters higher in the call stack are executed before code in your **finally** block is executed.

Note: Exception filters are supported by Microsoft Intermediate Language (MSIL) and Visual Basic .NET.

For more information, see [How To: Use Impersonation and Delegation in ASP.NET 2.0](#).

8.7.6 Avoid Losing Impersonation Tokens

In .NET Framework 1.1, impersonation tokens did not automatically flow to newly created threads. This situation could lead to security vulnerabilities because new threads assume the security context of the process. In ASP.NET 2.0 applications you can now change this default behavior by configuring the ASPNET.config file in the **%Windir%\Microsoft.NET\Framework\{Version}** directory.

If you need to flow the impersonation token to new threads, set the **enabled** attribute to **true** on the **alwaysFlowImpersonationPolicy** element in the ASPNET.config file, as shown in the following example.

```
<configuration>
  <runtime>
    <alwaysFlowImpersonationPolicy enabled="true"/>
  </runtime>
</configuration>
```

If you need to prevent impersonation tokens from being passed to new threads programmatically, you can use the **ExecutionContext.SuppressFlow** method.

8.8 Parameter Manipulation

Parameters, such as those found in form fields, query strings, view state, and cookies, can be manipulated by attackers who usually intend to gain access to restricted pages or trick the application into performing an unauthorized operation.

The following recommendations help you avoid parameter manipulation vulnerabilities:

- **Do not make security decisions based on parameters accessible on the client-side.**
- **Validate all input parameters.**
- **Avoid storing sensitive data in ViewState.**
- **Encrypt ViewState if it must contain sensitive data.**

8.8.1 Do Not Make Security Decisions Based on Parameters Accessible on the Client-Side

Do not trust input parameters, especially when they are used to make security decisions at the server. Also, do not use clear text parameters for any form of sensitive data. Instead, store sensitive data on the server in a session store and use a session token to reference the items in the store.

8.8.2 Validate All Input Parameters

Validate all input parameters that come from form fields, query strings, cookies, and HTTP headers. The **System.Text.RegularExpressions.Regex** class helps validate input parameters. For example, the following code shows how to use this class to validate a name passed through a query string parameter. The same technique can be used to validate other forms of input parameter, such as from cookies or form fields. For example, to validate a cookie parameter, use **Request.Cookies** instead of **Request.QueryString**.

```
using System.Text.RegularExpressions;
private void Page_Load(object sender, System.EventArgs e)
{
    // Name must contain between 1 and 40 alphanumeric characters
    // together with (optionally) special characters such as apostrophes
    // for names such as D'Angelo.
    if (!Regex.IsMatch(Request.QueryString["name"], @"^[a-zA-Z'\s]{1,40}$"))
        throw new Exception("Invalid name parameter");
    // Use individual regular expressions to validate all other
    // query string parameters.
}
```

For more information, see [How To: Protect from Injection Attacks in ASP.NET](#) and [How To: Use Regular Expressions to Constrain Input in ASP.NET](#).

8.8.3 Avoid Storing Sensitive Data in ViewState

Avoid storing sensitive data in ViewState. ViewState is not designed for sensitive data, and protecting it with encryption adds to performance overhead. If you need to manage sensitive data, maintain it on the server; for example, maintain it in session state.

If your ViewState does contain sensitive data, you should consider protecting it against eavesdropping by enabling ViewState encryption as described in the next section.

8.8.4 Encrypt ViewState if It Must Contain Sensitive Data

Using SSL protects ViewState while it is passed over the network between server and browser, but it does not stop it being viewed and modified on the user's computer.

To prevent ViewState from being viewed on the user's computer (and over the network), use ASP.NET ViewState encryption. Do not use it if your ViewState does not contain sensitive data because encryption significantly adds to the size of the ViewState and this impacts performance.

For more information about protecting ViewState, see [How To: Configure the Machine Key in ASP.NET 2.0](#).

8.8.5 Additional Considerations

In addition to the preceding guidelines, consider the following.

- **Use Page.ViewStateUserKey to counter one-click attacks.**

If you authenticate your callers and use ViewState, set the **Page.ViewStateUserKey** property in the **Page_Init** event handler to prevent one-click attacks.

```
void Page_Init (object sender, EventArgs e)
{
    ViewStateUserKey = Session.SessionID;
}
```

Set the property to a value you know is unique to each user, such as a session ID, user name, or user identifier.

A one-click attack occurs when an attacker creates a Web page (.htm or .aspx) that contains a hidden form field named `__VIEWSTATE` that is already filled with ViewState data. The ViewState can be generated from a page that the attacker had previously created, such as a shopping cart page with 100 items. The attacker lures an unsuspecting user into browsing to the page, and then the attacker causes the page to be sent to the server where the ViewState is valid. The server has no way of knowing that the ViewState originated from the attacker. ViewState validation and HMACs do not counter this attack because the ViewState is valid and the page is executed under the security context of the user.

By setting the **ViewStateUserKey** property, when the attacker browses to a page to create the ViewState, the property is initialized to his or her name. When the legitimate user submits the page to the server, it is initialized with the attacker's name. As a result, the ViewState HMAC check fails and an exception is generated.

Note: This attack is usually not an issue for anonymously browsed pages (where no user name is available), because this type of page should make no sensitive transactions.

8.9 Sensitive Data

Sensitive data includes application configuration details (for example, connection strings and service account credentials) and application-specific data (for example, customer credit card numbers). The following recommendations help to reduce risk when you handle sensitive data:

- **Avoid plaintext passwords in configuration files.**
- **Use platform features to manage keys where possible**
- **Do not pass sensitive data from page to page**
- **Protect sensitive data over the wire**
- **Do not cache sensitive data**

8.9.1 Avoid Plaintext Passwords in Configuration Files

The <sessionState> and <identity> elements in the Machine.config and Web.config files have **userName** and **password** attributes. If you store credentials in these sections, encrypt them by using one of the protected configuration providers.

For more information, see [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](#) and [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](#).

8.9.2 Use Platform Features to Manage Keys Where Possible

Use platform features where possible to avoid managing keys yourself. For example, by using DPAPI, the encryption key is derived from an account's password, so Windows handles this for you.

8.9.3 Do Not Pass Sensitive Data from Page to Page

Avoid using any of the client-side state management options, such as ViewState, cookies, query strings, or hidden form-field variables, to store sensitive data. The data can be tampered with and viewed in clear text. Use server-side state management options, such as a SQL Server database to help protect data exchange.

8.9.4 Protect Sensitive Data Over the Wire

Consider where items of sensitive data, such as credentials and application-specific data, are transmitted over a network link. If you need to send sensitive data between the Web server and browser, consider using SSL. If you need to protect server-to-server communication, such as between your Web server and database, consider IPSec or SSL.

8.9.5 Do Not Cache Sensitive Data

If your page contains data that is sensitive, such as a password, credit card number, or account status, the page should not be cached. Output caching is off by default.

8.10 Session Management

There are two main factors that you should consider to help protect session state. First, ensure that the session token cannot be used to gain access to sensitive pages where secure operations are performed or to gain access to sensitive items of data. Second, if the session data contains sensitive items, you must protect the session data, including the session store.

The following recommendations help you build secure session management:

- **Do not rely on client-side state management options.**
- **Protect your out-of-process state service.**
- **Protect SQL Server session state.**

8.10.1 Do Not Rely on Client-Side State Management Options

Avoid using any of the client-side state management options, such as view state, cookies, query strings, or hidden form fields, to store sensitive data. The information can be tampered with or seen in clear text. Use server-side state management options, for example, a database, to store sensitive data.

8.10.2 Protect Your Out-of-Process State Service

If you use **mode=StateServer** on the <sessionState> element, use the following recommendations to help secure session state:

- **Use a least-privileged account to run the state service.** By default, the state service runs using the Network Service local, least-privileged account. You should not need to change this configuration.
- **Protect the channel.** If the state service is located on a remote server and you are concerned about the network eavesdropping threat, protect the channel to the remote state store by using IPSec to ensure the user state remains private and unaltered.

- **Consider changing the default port.** The ASP.NET state service listens on port 42424. To avoid using this default, well-known port, you can change the port by editing the following registry key:

HKLM\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters

The port number is defined by the **Port** named value. If you change the port number in the registry, for example, to 45678, you must also change the connection string on the <sessionState> element, as follows:

```
stateConnectionString="tcpip=127.0.0.1:45678"
```

- **Encrypt the state connection string.** Because this element contains service connection details, you should encrypt it by using one of the protected configuration providers.

For more information, see [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](#) and [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](#).

8.10.3 Protect SQL Server Session State

If you use **mode="SQLServer"** on the <sessionState> element, use the following recommendations to help protect session state:

- **Use Windows authentication to the database.** This means that passwords are not sent over the network to the database. It also means you do not have to store user names and passwords in the database connection string.
- **Encrypt the <sessionState> section.** Because this element contains database connection details, you should encrypt it by using one of the protected configuration providers.

For more information, see [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](#) and [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](#).

- **Limit the application's login in the database.** Restrict the login in the database so that it can only be used to access the necessary state tables and the stored procedures used by ASP.NET to query the database. To do this, create a SQL Server login for your ASP.NET process account, and then map the login to a database user in the state store database. Assign the database user to a database role and grant execute permissions for the stored procedures that are provided with the ASPState database.
- **Protect the channel.** To protect sensitive session state over the network between the Web server and remote state store, protect the channel to the two servers using IPsec or SSL. This provides privacy and integrity for the session state data across the network. If you use SSL, you must install a server certificate on the database server.

8.11 Auditing and Logging

You should audit and log activity across the tiers of your application. Using logs, you can detect suspicious-looking activity. This frequently provides early indications of a full-blown attack and the logs help address the repudiation threat where users deny their actions. Log files may be required in legal proceedings to prove the wrongdoing of individuals. Generally, auditing is considered most authoritative if the audits are generated at the precise time of resource access and by the same routines that access the resource.

When implementing auditing and logging in ASP.NET applications:

- **Use health monitoring to log and audit events.**
- **Instrument for user management events.**
- **Instrument for unusual activity.**
- **Instrument for significant business operations.**
- **Consider using an application-specific event source.**
- **Protect audit and log files.**

8.11.1 Use Health Monitoring to Log and Audit Events

ASP.NET version 2.0 introduces a health monitoring feature that you should use to log and audit events. By default, health monitoring is enabled for ASP.NET version 2.0 applications and all Web infrastructure error events (inheriting from **System.Web.Management.WebErrorEvent**) and all audit failure events (inheriting from **System.Web.Management.WebFailureAuditEvent**) are written to the event log. The default configuration is defined in the `<healthMonitoring>` element in the machine-level Web.config file. To audit additional security related events, you create custom event types by deriving from one of the built-in types.

The health monitoring feature has built-in providers that allow you to log events in an e-mail message (**SimpleMailWebEventProvider**, **TemplatedMailWebEventProvider**), to SQL Server (**SqlWebEventProvider**), to the event log (**EventLogWebEventProvider**), as ASP.NET trace output (**TraceWebEventProvider**), or to the Windows Management Instrumentation (WMI) Web event provider (**WMIWebEventProvider**). You can configure health monitoring in the machine or application Web.config file to modify the events that are logged and the way in which they are logged.

For more information, see [How To: Use Health Monitoring in ASP.NET 2.0](#).

8.11.2 Instrument for User Management Events

Instrument your application and monitor user management events such as password resets, password changes, account lockout, user registration, and authentication events. Doing this helps you to detect and react to potentially suspicious behavior. It also enables you to gather operations data; for example, to track who is accessing your application and when user account passwords need to be reset.

By default, ASP.NET health monitoring records all **WebFailureAuditEvents**, which ASP.NET raises when user authentication fails in forms authentication with the membership system. ASP.NET also raises a **WebFailureAuditEvent** when authorization is not granted to a resource protected by file authorization and URL authorization.

For more information about auditing password changes and account lockout events, see [How To: Instrument ASP.NET 2.0 Applications for Security](#).

8.11.3 Instrument for Unusual Activity

Instrument your application and monitor events that might indicate unusual or suspicious activity. This enables you to detect and react to potential problems as early as possible. Unusual activity might be indicated by:

- Replays of old authentication tickets.
- Too many login attempts over a specific period of time.

Replays of old authentication tickets automatically raise an **ExpiredTicketFailure** event. To raise an event for too many login attempts, if you are using the membership system and the **Login** controls, you can write an event handler for the **LoginError** event of the **Login** control. In the handler, call **Membership.GetUser(Login1.UserName).IsLockedOut** to determine if there have been too many login attempts for the user. You can then raise a custom event to indicate that the account has been locked out.

For more information about auditing password changes and account lockout events, see [How To: Instrument ASP.NET 2.0 Applications for Security](#).

8.11.4 Instrument for Significant Business Operations

Use ASP.NET health monitoring to track significant business operations. For example, instrument your application to record access to particularly sensitive methods and business logic. To do this, create a custom event class that inherits from **System.Web.Management.WebSuccessAuditEvent** or **System.Web.Management.WebFailureAuditEvent** and raise that event in the appropriate methods.

For more information see [How To: Instrument ASP.NET 2.0 Applications for Security](#).

8.11.5 Consider Using an Application-specific Event Source

By default, the **EventLogWebEventProvider** writes to the application event log by using an event source named "ASP.NET xxxxxx", where xxxxxx is the .NET Framework version number. This is not configurable. If you want events generated by the ASP.NET health monitoring feature to use an application-specific event source, you must create a custom event provider (inherit from **EventLogWebEventProvider**) and override the **ProcessEvent** method to write to the event log (by calling **EventLog.Write**) using the desired event source.

When using a custom event source, you need to create the event source at installation time when administrator privileges are available. If you are unable to create new event sources at installation time, and you are in deployment, the administrator should manually create new event source entry beneath the following registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\<LogName>

Note: You should not grant write permission to the ASP.NET process account (or any impersonated account if your application uses impersonation) on the

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog registry key. If you allow write access to this key and the account is compromised, the attacker can modify any log-related setting, including access control to the log, for any log on the system.

8.11.6 Protect Audit and Log Files

Protect audit and log files using Windows ACLs and restrict access to the log files. If you log events to SQL Server or to some custom event sink, use appropriate access controls to limit access to the event data. For example, grant write access to the account or accounts used by your application, grant full control to administrators, and read-only access to operators.

This makes it more difficult for attackers to tamper with log files to cover their tracks. Minimize the number of individuals who can manipulate the log files. Authorize access only to highly trusted accounts such as administrators.

8.11.7 Additional Considerations

In addition to the preceding guidelines, consider the following measures for auditing and logging:

- **Log application events on a separate, protected server.** This helps to ensure that logs cannot be tampered by attackers.
- **Assign appropriate permissions to the log files.** Logs should be written by a process with write permission only. Logs should be read by users with administrative access.
- **Log application events with sufficient details.** Provide sufficient detail to permit reconstruction of system activity.
- **Use performance counters for high volume, per-request events.** This helps to minimize the impact on performance.

8.12 Deployment Considerations

To avoid introducing vulnerabilities when you deploy your application into a production environment:

- **Use a least-privileged account for running ASP.NET applications.**
- **Encrypt configuration sections that store sensitive data.**
- **Consider your key storage location.**
- **Block protected file retrieval by using HttpForbiddenHandler.**
- **Configure the MachineKey to use the same keys on all servers in a Web farm.**
- **Lock configuration settings to enforce policy settings.**

8.12.1 Use a Least-Privileged Account for Running ASP.NET Applications

On IIS 5, ASP.NET Web applications run by default using the least privileged ASPNET account. On IIS 6, they run inside an application pool using the least privileged Network Service account. If you need to use a custom service account to run your ASP.NET Web application, create a least-privileged account. You might need to create a custom service account to isolate your application from other applications on the same server or to be able to audit each application separately.

To create a least privileged account

1. Create a local or domain Windows account.
2. Run the following `Aspnet_regiis.exe` command to assign the relevant ASP.NET permissions to the account:

```
aspnet_regiis.exe -ga machineName\userName
```

On Windows Server 2003, running the `Aspnet_regiis.exe -ga` command will add the account to the IIS_WPG group. The IIS_WPG group provides the **Log on as a batch job** user right and ensures that the necessary file system permissions are granted.

Note: With ASP.NET Beta 2, the `Aspnet_regiis.exe -ga` command does not add the account to the IIS_WPG group. However, it will do so prior to the final release of ASP.NET 2.0.

3. Use the Local Security Policy tool to grant the Windows account the **Deny logon locally** user right. This reduces the privileges of the account and prevents anyone logging onto Windows locally with the account.
4. Use IIS Manager to create an application pool running under the new account's identity and assign your ASP.NET application to the pool.

For more information, see [How To: Create a Service Account for an ASP.NET 2.0 Application](#).

8.12.2 Encrypt Configuration Sections That Store Sensitive Data

In ASP.NET version 1.1, you could use the `Aspnet_setreg.exe` tool to encrypt certain elements in the `Web.config` and `Machine.config` files that contain credentials. This tool replaces the credentials in the configuration file with a pointer to a registry key that holds the DPAPI encrypted data. In ASP.NET version 2.0, you can use either the DPAPI or RSA protected configuration providers to encrypt specific sections that contain sensitive data. The sections that usually contain sensitive information that you need to encrypt include `<appSettings>`, `<connectionStrings>`, `<identity>`, and `<sessionState>`.

To encrypt the `<connectionStrings>` section by using the DPAPI provider with the machine-key store (the default configuration), run the following command from a command window:

```
aspnet_regiis -pe "connectionStrings" -app "/MachineDPAPI" -prov  
"DataProtectionConfigurationProvider"
```

- **-pe:** Specifies the configuration section to encrypt.
- **-app:** Specifies your Web application's virtual path. If your application is nested, you need to specify the nested path from the root directory; for example, `/test/aspnet/MachineDPAPI`.
- **-prov:** Specifies the provider name.

If you need to encrypt configuration file data on multiple servers in a Web farm, you should use RSA protected configuration provider because of the ease with which you can export RSA key containers.

For more information, see [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](#) and [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](#).

8.12.3 Consider Your Key Storage Location

The DPAPI and RSA protected configuration providers used to encrypt sensitive data in configuration files can use either machine stores or user stores for key storage. You can either store the key in the machine store and create an ACL for your specific application identity or store the key in a user store. In the latter case, you need to load the user account's profile to access the key.

Use machine-level key storage when:

- Your application runs on its own dedicated server with no other applications.
- You have multiple applications on the same server that run using the same identity and you want those applications to be able to share sensitive information and the same encryption key.

Use user-level key storage if you run your application in a shared hosting environment and you want to ensure that your application's sensitive data is not accessible to other applications on the server. In this scenario, each application should have a separate identity so they all have their own individual and private key stores.

8.12.4 Block Protected File Retrieval by Using `HttpForbiddenHandler`

To prevent protected files being downloaded over HTTP, map them to the ASP.NET `HttpForbiddenHandler`. HTTP handlers are located in the machine-level `Web.config` file beneath the `<httpHandlers>` element. HTTP handlers are responsible for processing Web requests for specific file extensions.

Many well-known file extensions are already mapped to `System.Web.HttpForbiddenHandler`. For .NET Framework resources, if you do not use a particular file extension, map the extension to `System.Web.HttpForbiddenHandler` in the machine-level `Web.config` file, as shown here.

```
<add verb="*" path="*.asmx" type="System.Web.HttpForbiddenHandler" />
```

This example assumes that your Web server does not host Web services, so the `.asmx` file extension is mapped to `System.Web.HttpForbiddenHandler`. If a client requests a path that ends with `.asmx`, ASP.NET returns a message that says, "This type of page is not served." If your application uses other files with extensions not already mapped, also map these to `System.Web.HttpForbiddenHandler`.

8.12.5 Configure the `MachineKey` to Use the Same Keys on All Servers in a Web Farm

If you deploy your application in a Web farm, you must ensure that the configuration files on each server share hashing and encryption keys. These are used by ASP.NET to protect ViewState and forms authentication tickets. Manually generated, common keys are required because you cannot guarantee which server will handle successive requests.

With manually generated key values, the `<machineKey>` settings should be similar to the following example.

```
<machineKey validationKey="21F090935F6E49C2C797F69BBAAD8402ABD2EE0B667A8B4  
4EA7DD4374267A75D7AD972A119482D15A4127461DB1DC347C1A63AE5F1CCFAACFF1B72A7F0A2  
81B"  
decryptionKey="261F793EB53B761503AC445E0CA28DA44AA9B3CF06263B77"  
decryption="3DES"  
validation="SHA1" />
```

For more information about how to manually generate keys, see [How To: Configure MachineKey in ASP.NET 2.0](#).

8.12.6 Lock Configuration Settings to Enforce Policy Settings

To prevent individual application's from being able to override configuration settings with their own Web.config files, apply configuration settings centrally in the machine-level Web.config file located in the following directory: %Windir%\Microsoft.NET\Framework\{version}\CONFIG.

To lock the configuration settings for all Web applications on a Web server, place the configuration settings inside a **<system.web>** element nested within a **<location>** element and set the **allowOverride** attribute to **false**.

The following example enforces the use of Windows authentication for all Web applications on the server.

```
<location allowOverride="false">
  <system.web>
    <authentication mode="Windows"/>
  </system.web>
</location>
```

If you need to apply and lock settings for a specific Web application, use the path attribute on the **<location>** element to identify the Web application as shown here.

```
<location path="Default Web Site/VDirName">
  <system.web>
    <authentication mode="Windows"/>
    <identity impersonate="false"/>
  </system.web>
</location>
```

If you specify the path, it must be fully qualified and include the Web site name and virtual directory name.

8.13 Communication Security

When passing sensitive data, such as credentials or application-specific data, across networks, you need to consider the security of the communication channel:

- **Consider SSL vs. IPsec.**
- **Optimize pages that use SSL.**

8.13.1 Consider SSL vs. IPsec

If your servers are not inside a physically secure data center where the network eavesdropping threat is considered insignificant, you need to use an encrypted communication channel to protect data sent between servers. SSL and IPsec can both be used to help protect communication between servers by encrypting traffic. Use SSL when you need granular channel protection for a particular application instead of for all applications and services running on a computer.

Use IPsec to help protect the communication channel between two servers and to restrict which computers can communicate with one another. For example, you can help protect a database server by establishing a policy that permits requests only from a trusted client computer such as an application or Web server. You can also restrict communication to specific IP protocols and TCP/UDP ports.

8.13.2 Optimize Pages That Use SSL

Using SSL is expensive. Use SSL only for pages that require it. This includes pages that contain or capture sensitive data, such as pages that accept credit card numbers and passwords. Use SSL only if the following conditions are true:

- You want to encrypt the page data.
- You want to guarantee that the server to which you send the data is the server that you expect.

For pages where you must use SSL, follow these guidelines:

- Make the page size as small as possible.
- Avoid using graphics that have large file sizes. If you use graphics, use graphics that have smaller file sizes and resolution. Or, use graphics from a site that is not secure. However, when you use graphics from a site that is not secure, Web browsers display a dialog box that asks the user if the user wants to display the content from the site that is not secure.

9.0 Appendix A

Include files in .Net

An ASP.NET include, or server-side include, is a plain HTML, HTML/ASP.NET, or ASP.NET-only file that sits outside of a parent Web page. The power of a simple include file is tremendous—it allows to re-use content that will appear on multiple pages throughout site. Instead of making the same change 100 times to update 100 pages, we could use an include file, update the include file once and have the change populated across all 100 pages.

When to use ASP.NET includes

Any piece of content that will appear on multiple pages should be an include file, especially on larger sites. As you become familiar with them, you will get creative in their usage. Some Common uses are site navigation, copyright info, footer links, and “News” sections, to name a few.

You should use a small amount of discretion, however, when deciding when and where to use them. If the content is not reused on multiple pages or you don’t have a slick dynamic script that requires them, you don’t need to use includes. Include files put a minor amount of extra strain on the server. When they are used correctly, the benefits far outweigh any performance issues. If you go crazy and start using them for every small bit of code, though, you may notice a difference in your site’s load time.

Using an ASP.NET include

Let’s assume you have a 100 page website. On the bottom of all 100 pages you have secondary navigation, copyright information, and a link to your privacy policy. If we decide to add an additional link to the footer navigation one day, it would really be arduous to make the update 100 times by copying and pasting code into every file. Luckily, if we use ASP.NET includes, we only need to make the update once.

Here’s an example of a basic ASP.NET file (we’ll call it index.aspx):

```
<Script Runat="Server">
'Any ASP.NET code would go here
</Script>
<html>
  <head>
    <title>My ASP.NET Home Page</title>
  </head>
  <body>
    Some text goes here
  </body>
</html>
```

Here’s our ASP.NET include file to include the footer links and copyright information (we’ll call it inc_footer.aspx):

```
<p>
Copyright 2004-2005 mysite.com. All rights reserved. <br/>
<a href="index.aspx">Home</a> |
<a href="about-us.aspx">About Us</a> |
<a href="services.aspx">Services</a> |
<a href="contact.aspx">Contact</a> |
<a href="privacy-policy.aspx">Privacy Policy
</p>
```

To include our copyright and footer links in the home, about, services, contact, and every other page, we only need to add one line of code to each of them:

```
<!--#include file="inc_footer.aspx"-->
```

index.aspx now looks like:

```
<Script Runat="Server">
' Any ASP.NET code would go here
</Script>
<html>
  <head>
    <title>My ASP.NET Home Page</title>
  </head>
  <body>
    <p>Some text goes here</p>
    <!--#include file="inc_footer.aspx"-->
  </body>
</html>
```

That's it! For every other page in the site, you only need to add one line of code to insert the footer content. A new year rolls around? No problem. Open up inc_footer.aspx, change the year, upload it, and the update is made across every page in your site.

Naming your ASP.NET include files

Technically, you can name your ASP include file whatever you want, as long as it ends in ASPX or INC. I recommend giving your include files a prefix of "inc-" (as in inc-filename.aspx) or "inc_" (as in inc_filename.aspx). You might even consider placing them in their own folder. I find that this helps in managing files, especially when working on a bigger site.

I strongly recommend giving your include files an **.ASPX extension only**. The reason is this— if someone snooping around in your site guesses the correct filename for your include file and the extension ends in .inc, the full source code will be displayed in the browser (the browser sees a ".inc" file as plain text, not a Web document). This is a huge security flaw, especially when you move into more complex programming (can reveal database connection strings, passwords, etc.). If the snooper guesses correctly but your include file ends in .ASPX, any ASP.NET code will be processed first (the server recognizes it as a server-side programming file), and the person will only see the OUTPUT of the code, not the source code itself.

How To Dynamically Include Files in ASP.NET

This article demonstrates how to dynamically include HTML and client-side scripts in .aspx pages.

Because ASP.NET applications are compiled and run before they are sent to the client, you cannot use a variable in place of a file name in a server-side include file (such as `<!-- #include PathType = FileName -->`). However, you can use the **Response** or **StreamReader** object to write the include file to the HTTP content stream.

This article demonstrates how to create an .aspx page that reads a file with static HTML and/or client-side script code and writes the file's content to the browser.

MORE INFORMATION

In ASP.NET, the **Response** object provides a new method named **WriteFile**. You can use the **WriteFile** method to write the specified file directly to an HTTP content output stream.

If you only want to write the content of a file to the browser, you can accomplish this in just one statement. If you want to manipulate the file before you send it to the browser, refer to the [References](#) section for information about basic file input/output in .NET.

In ASP.NET, you can either write inline code or write code in the code-behind module. This article presents an inline code sample that opens a file and writes the file's content to the browser.

Steps to Create the Sample

- Open Microsoft Visual Studio .NET.
- From the **File** menu, point to **New**, and then click **Project**.
- In the **New Project** dialog box, click **Visual Basic Projects** under **Project Types**. Under **Templates**, click **ASP.NET Web Application**.
- Switch to the HTML code editor for the .aspx page that is created by default. Replace the existing code with the following code:
- ```
<%@ Page Language="vb" AutoEventWireup="false"%> <html> <body> <% Response.WriteFile("Yourfile.inc") %> </body> </html>
```
- Replace "Yourfile.inc" in the **Response.WriteFile** statement with the name of an include file that contains some HTML or client-side script.
- Add "Yourfile.inc" to the project.
- Browse to the .aspx file. Note that the content of your file is written to the browser.

### Troubleshooting

- Server-side code in the dynamically included file is displayed on the client browser.
- The dynamically included file may contain any client-side code, including HTML and JavaScript. If that file contains any server-side code, the server-side code is sent to the client browser as plain text and is visible if you view the source of the page that is displayed in the browser. Note that ASP.NET does not process server-side script in the dynamically included file. This is because all of the ASP.NET code has already run before it includes the file; thus, the server does not return to read anything for server-side processing again.
- If you use `Response.Write` or `Response.WriteFile` statements in a code-behind module, these statements write the information before any HTML tags. The same behavior occurs if you use inline `<SCRIPT>` tags with the `RUNAT="Server"` attribute.
- Because the code-behind modules are compiled first, all of the output that is generated by `Response.Write`, `Response.WriteFile`, or inline server-side `<SCRIPT>` tags appears before any HTML tags when the HTML output is sent to the browser. This problem does not occur when you use `Response.Write` statements in classic ASP-style tags.

## 10.0 Appendix B

### ASP.NET 2.0 Web.Config

Web.config files can appear in multiple directories in ASP.NET applications. Each Web.config file applies configuration settings to its own directory and to all the child directories below it. Settings in child directories can optionally override or modify settings that are specified in parent directories. The root of the ASP.NET configuration hierarchy is the *systemroot*\Microsoft.NET\Framework\versionNumber\CONFIG\Web.config file, which includes settings that apply to all ASP.NET applications that run a specific version of the Microsoft .NET Framework. Because each ASP.NET application inherits default configuration settings from the root Web.config file, you only need to create Web.config files for settings that override the default settings.

### Case-Sensitivity

Because tags must be well-formed XML, the tags, subtags, and attributes are case-sensitive. Tag names and attribute names are camel-cased, which means that the first character of a tag name is lowercase and the first letter of any subsequent concatenated word or words is uppercase. In most cases, string attribute values are Pascal-case, which means that the first character is uppercase and the first letter of any subsequent concatenated word or words is uppercase. Exceptions are **true** and **false**, which are always lowercase.

### Custom Sections

The ASP.NET configuration infrastructure makes no assumptions about the types of configuration data that the infrastructure supports. Configuration section-handler classes process all Web.config data. You can use the predefined configuration section handlers that are supplied with the .NET Framework, or you can create your own handlers to process custom configuration data.

### Configuring Specific Files and Subdirectories

Configuration settings can be applied to specific resources by using a `<location>` tag with an appropriate **path** attribute. The **path** attribute can be used to identify a specific file or child directory to which unique configuration settings apply. Only one file path can be used in the **path** attribute. The **path** attribute can start with the `~/` shortcut which indicates the application root.

For example, the following example configuration file specifies settings at three levels:

- Settings that apply to the current directory and all child directories (everything contained within the top `<configuration>` tag).
- Settings that apply to the `Sub1` child directory (everything contained within the `<location>` tag with a path attribute set to `Sub1`).
- Settings that apply to the `Sub2` child directory (everything contained within the `<location>` tag with a path attribute set to `Sub2`).

```

<configuration>
 <system.web>
 <sessionState cookieless="true" timeout="10"/>
 </system.web>

 <!-- Configuration for the "sub1" subdirectory. -->
 <location path="sub1">
 <system.web>
 <httpHandlers>
 <add verb="*" path="sub1" type="Type1"/>
 <add verb="*" path="sub1" type="Type2"/>
 </httpHandlers>
 </system.web>
 </location>

 <!-- Configuration for the "sub1/sub2" subdirectory. -->
 <location path="sub1/sub2">
 <system.web>
 <httpHandlers>
 <add verb="*" path="sub1/sub2" type="Type3"/>
 <add verb="*" path="sub1/sub2" type="Type4"/>
 </httpHandlers>
 </system.web>
 </location>
</configuration>

```

There are number of important settings that can be stored in the configuration file. Here are some of the **most frequently used configurations**, stored conveniently inside Web.config file.

1. [Database](#) connections
2. Session States
3. Error Handling
4. Security

## Database Connections:

The most important configuration data that can be stored inside the web.config file is the database connection string. Storing the connection string in the web.config file makes sense, since any modifications to the database configurations can be maintained at a single location. As otherwise we'll have to keep it either as a class level variable in all the associated source files or probably keep it in another class as a public static variable.

But if this is stored in the Web.config file, it can be read and used anywhere in the program. This will certainly save us a lot of alteration in different files where we used the old connection.

Let's see a small example of the connection string which is stored in the web.config file.

```
<configuration>
 <connectionStrings>
 <add name="MyConnection" connectionString="data source=Palmetto;Initial Catalog=gdfSurvey;User
 ID=tillson;Password=XX" providerName="System.Data.SqlClient"/> </connectionStrings>
 </configuration>
```

Let's see how we access the connection string from our Asp .net web application.

You can reference this directly from code using:

**[C#]**

```
string connStr = ConfigurationManager.ConnectionStrings["MyConnection"].ConnectionString;
```

**[VB]**

```
Dim connStr As String = ConfigurationManager.ConnectionStrings("MyConnection").ConnectionString
```

The small code snippet above is all that is needed to access the value stored inside the Web.config file.

## Session States:

Session in Asp .net web application is very important. As we know that HTTP is a stateless protocol and we need session to keep the state alive. Asp .net stores the sessions in different ways. By default the session is stored in the asp .net process. You can always configure the application so that the session will be stored in one of the following ways.

### Session State Service

There are two main advantages of using the State Service. First the state service is not running in the same process as the asp .net application. So even if the asp .net application crashes the sessions will not be destroyed. Any advantage is sharing the state information across a Web garden (Multiple processors for the same computer).

Let's see a small example of the Session State Service.

```
<sessionState mode="StateServer" stateConnectionString="tcpip=127.0.0.1:55455" sqlConnectionString="data
source=127.0.0.1;user id=sa;password=" cookieless="false" timeout="20"/>
```

The attributes are self explanatory but I will go over them.

**mode:** This can be StateServer or SqlServer. Since we are using StateServer we set the mode to StateServer.

**stateConnectionString:** connectionString that is used to locate the State Service.

**sqlConnectionString:** The connection String of the sql server database.

**cookieless:** Cookieless equal to false means that we will be using cookies to store the session on the client side.

## SQL Server

The final choice to save the session information is using the Sql Server 2000 database. To use Sql [Server](#) for storing session state you need to do the following:

1) Run the InstallSqlState.sql script on the Microsoft SQL Server where you intend to store the session.

Your web.config settings will look something like this:

```
<sessionState mode = "SqlServer" stateConnectionString="tcpip=127.0.0.1:45565" sqlConnectionString="data source='SERVERNAME';user id=sa;password=' ' cookiesless='false' timeout='20' />
```

SQL Server lets you share session state among the processors in a Web garden or the servers in a Web farm. Apart from that you also get additional space to store the session. And after that you can take various actions on the session stored.

The downside is SQL Server is slow as compared to storing session in the state in process. And also SQL Server cost too much for a small company.

## InProc:

This is another Session State. This one is mostly used for development purposes. The biggest advantage of using this approach is the applications will run faster when compared to other Session state types. But the disadvantage is Sessions are not stored when there is any problem that occurs with the application, when there is a small change in the files etc., Also there could be frequent loss of session data experienced.

## Error Handling:

See [Exception Management: Section 8.6.2](#).

## Security:

The most critical aspect of any application is the security. Asp.net offers many different types of security method which can be used depending upon the condition and type of security you need.

## No Authentication:

No Authentication means "No Authentication" :), meaning that Asp.net will not implement any type of security.

## Windows Authentication:

The Windows authentication allows us to use the [windows user](#) accounts. This provider uses IIS to perform the actual authentication, and then passes the authenticated identity to your code. If you like to see that what windows user is using the Asp.net application you can use:

### User.Identity.Name;

This returns the [DOMAIN](#)UserName of the current user of the local machine.

## Passport Authentication:

Passport Authentication provider uses [Microsoft's](#) Passport service to authenticate users. You need to purchase this service in order to use it.

## Forms Authentication:

Forms Authentication uses HTML forms to collect the user information and then it takes required actions on those HTML collected values.

In order to use Forms Authentication you must set the Anonymous Access checkbox checked. Now we need that whenever user tries to run the application he/she will be redirected to the login page.

```
<authentication mode="Forms">
<forms loginUrl = "frmLogin.aspx" name="3345C" timeout="1"/>
</authentication>
<authorization>
<deny users="?" />
</authorization>
```

As you can see we set the Authentication mode to "Forms". The forms loginUrl is the first page being displayed when the application is run by any user.

The authorization tags has the deny users element which contains "?", this means that full access will be given to the authenticated users and none access will be given to the unauthenticated users. You can replace "?" with "\*" meaning that all access is given to all the users no matter what.

## 11.0 References

"MSDN: .NET Framework Developer's Guide: Design Guidelines for Developing Class Libraries" Microsoft, 2005.

[http://msdn.microsoft.com/en-us/library/ms229042\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/ms229042(VS.80).aspx)

"MSDN: .NET Framework Developer's Guide: Design Guidelines for Class Library Developers" Microsoft, 2005.

<http://msdn.microsoft.com/en-us/library/czefa0ke.aspx>

"MSDN: Visual Basic: Samples: Visual Basic Coding Conventions" Microsoft, 2005.

[http://msdn.microsoft.com/en-us/library/h63fsef3\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/h63fsef3(VS.80).aspx)

"MSDN: .NET Framework Developer's Guide: Secure Coding Guidelines" Microsoft, 2005.

[http://msdn.microsoft.com/en-us/library/d55zzx87\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/d55zzx87(VS.80).aspx)

"Standard ECMA-334 C# Language Specification 4th edition (June 2006)"

<http://www.ecma-international.org/publications/standards/Ecma-334.htm>

"DotNetSpider.com: C# Coding Standards and Best Programming Practices"

<http://www.dotnetspider.com/tutorials/BestPractices.aspx>

"MSDN: Security Guidelines: ASP.NET 2.0"

[http://msdn.microsoft.com/en-us/library/ms998258.aspx#pagguidelines0001\\_indexofguidelines](http://msdn.microsoft.com/en-us/library/ms998258.aspx#pagguidelines0001_indexofguidelines)

"MSDN: ASP.NET Technical Articles: Simulating include Files with an ASP.NET Server Control"

<http://msdn.microsoft.com/en-us/library/aa479009.aspx>

"Microsoft Help and Support: How To Dynamically Include Files in ASP.NET"

<http://support.microsoft.com/kb/306575>

"4 Guys From Rolla.com: Accessing Common Code, Constants, and Functions in an ASP.NET Project"

<http://aspnet.4guysfromrolla.com/articles/122403-1.aspx>

"MSDN: Visual Web Developer: How to: Create Web.config Files"

[http://msdn.microsoft.com/en-us/library/k8x4ket8\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/k8x4ket8(VS.80).aspx)

"MSDN: ASP.NET Configuration Editing ASP.NET Configuration Files"

[http://msdn.microsoft.com/en-us/library/ackhksh7\(vs.80\).aspx](http://msdn.microsoft.com/en-us/library/ackhksh7(vs.80).aspx)

"MSDN: ASP.NET Configuration: How to: Create Custom Configuration Sections Using ConfigurationSection"

[http://msdn.microsoft.com/en-us/library/2tw134k3\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/2tw134k3(VS.80).aspx)

"MSDN: ASP.NET Configuration: Configuring Specific Files and Subdirectories"

[http://msdn.microsoft.com/en-us/library/6hbkh9s7\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/6hbkh9s7(VS.80).aspx)

"Microsoft .net Framework SDK: Managing ASP.NET Applications: Using the Management API"

<http://quickstarts.asp.net/QuickStartv20/aspnet/doc/management/mgmtapi.aspx#encrypt>

"CoderSource.net: Asp .net Web.config Configuration File"

[http://www.codersource.net/asp\\_net\\_web\\_configuration\\_file.html](http://www.codersource.net/asp_net_web_configuration_file.html)

"InformIT: Creating ASP.NET Applications"

<http://www.informit.com/articles/printerfriendly.aspx?p=27315>

"CERT Secure Coding Standards"

<https://www.securecoding.cert.org/confluence/display/cplusplus/MS07-CPP.+Detect+and+remove+dead+code>



NYS OFFICE OF GENERAL SERVICES

*Serving New York*

**DIVISION OF FINANCIAL ADMINISTRATION**

**REQUEST FOR QUOTE**

**Group Number:** 73012

**Class Codes:** 84, 80, 82, 83, 43, 86, 81

**RFQ Number:** 1768

**Title:** IT Services to Develop a Web-Based Application “eHBITS” for the Hourly-Based IT Services (HBITS) Contract

**Contract Period:** As Specified

**Due Date:** August 2, 2012 at 2:00 PM

**Address Inquiries To:** Kathleen Gallagher  
NYS Office of General Services  
40<sup>TH</sup> Floor, Corning Tower  
Empire State Plaza  
Albany, NY 12242  
[Kathleen.Gallagher@ogs.ny.gov](mailto:Kathleen.Gallagher@ogs.ny.gov)

**TABLE OF CONTENTS**

<b>1. GENERAL INFORMATION .....</b>	<b>3</b>
A. EXECUTIVE SUMMARY .....	3
B. DESIGNATED CONTACT .....	3
C. INTENT TO QUOTE .....	4
D. SECTION HEADERS.....	4
<b>2. ADMINISTRATIVE INFORMATION .....</b>	<b>4</b>
A. RFQ QUESTIONS AND CLARIFICATIONS .....	4
B. KEY EVENTS.....	5
C. CONTRACT PERIOD .....	5
D. PRICE.....	5
E. SUBMISSION DOCUMENTS.....	7
F. BID DELIVERY.....	7
G. METHOD OF AWARD .....	7
H. PAYMENT AND FINAL ACCEPTANCE (RETAINAGE) .....	8
I. ELECTRONIC PAYMENTS .....	8
<b>3. SCOPE OF WORK.....</b>	<b>8</b>
A. IN-SCOPE WORK.....	8
B. CONFIDENTIALITY .....	13
C. OUT-OF-SCOPE WORK .....	13
<b>4. CONTRACT CLAUSES AND REQUIREMENTS .....</b>	<b>14</b>
A. APPENDIX A.....	14
B. CONFLICT OF TERMS AND CONDITIONS .....	14
C. SUMMARY OF POLICY AND PROHIBITIONS ON PROCUREMENT LOBBYING .....	14
D. CONTRACTOR INSURANCE REQUIREMENTS .....	14
E. EMPLOYEE INFORMATION TO BE REPORTED BY CERTAIN CONSULTANT CONTRACTORS.....	16
F. TAX AND FINANCE CLAUSE.....	17
G. MWBE AND EEO REQUIREMENTS.....	17
H. NYS VENDOR FILE REGISTRATION .....	18
I. NYS STANDARD VENDOR RESPONSIBILITY QUESTIONNAIRE .....	19
J. GENERAL REQUIREMENTS .....	19
K. DISPUTE RESOLUTION.....	20
L. TERMINATION .....	20
M. SERVICES WARRANTY.....	20
N. INFORMATION SECURITY BREACH AND NOTIFICATION ACT .....	20
O. WORK IN THE UNITED STATES .....	21
P. ADDITIONAL PAYMENT PROVISION.....	21
Q. USE RESTRICTIONS AND INTELLECTUAL PROPERTY.....	21
R. CONSULTANT REPLACEMENTS .....	21
S. NYS PROCUREMENT RIGHTS.....	22

**Appendix A – Standard Clauses for New York State Contracts**

**Appendix B – Required Forms**

**Appendix C – Sample Service Agreement**

**Exhibit 1 – HBITS Process Flow and Selection Forms**

**Exhibit 2 – eHBITS PowerPoint Presentation**

**IMPORTANT NOTICE TO POTENTIAL BIDDERS:** Receipt of this request for quote document does not indicate that the Office of General Services has pre-determined your company's qualifications to receive a contract award. Such determination will be made after the request for quote due date and will be based on our evaluation of your quote submission compared to the specific requirements and qualifications contained in this request for quote document.

**1. GENERAL INFORMATION:**

**A. Executive Summary**

Since taking office in January, Governor Andrew M. Cuomo has committed his administration to implementing enterprise-wide changes that will utilize modern business practices in running New York State government. New Yorkers need a government in which they can take pride, and this comprehensive overhaul of operations will help accomplish that goal. This RFQ is a part of Governor Cuomo’s Procurement Transformation initiative to cut waste and improve government efficiency.

The Office of General Services (OGS) is soliciting quotes from New York State small businesses and certified Minority / Women owned Business Enterprises (M/WBEs) to develop a web-based application to implement the solutions identified in the Hourly-Based IT Services (HBITS) RFP.

This document will be referred to as the “eHBITS RFQ”. The scope of this RFQ can be found in Section 2.

**B. Designated Contacts**

In compliance with the Procurement Lobbying Law, the following individual has been designated as the PRIMARY contact for this procurement solicitation:

Kathleen Gallagher, Purchasing Agent  
NYS Office of General Services  
Financial Administration  
Corning Tower, 40<sup>th</sup> Floor, ESP  
Albany, NY 12242  
Email: [Kathleen.Gallagher@ogs.ny.gov](mailto:Kathleen.Gallagher@ogs.ny.gov)

The following SECONDARY Contact has been designated in case the Primary Contact is unavailable:

David Burmaster  
NYS Office of General Services  
Strategic Sourcing Team  
Corning Tower, 37<sup>th</sup> Floor, ESP  
Albany, NY 12242  
Email: [ITSProcurement@ogs.ny.gov](mailto:ITSProcurement@ogs.ny.gov)

**No inquiries or questions submitted via phone will be accepted.**

**C. Intent to Quote**

Potential bidders are strongly encouraged to indicate their intent to quote on RFQ #1768 – IT Services to Develop a Web-Based Application (eHBITS), by emailing Kathleen Gallagher at [Kathleen.Gallagher@ogs.ny.gov](mailto:Kathleen.Gallagher@ogs.ny.gov) by the deadline as stated in Section 2.B - Key Events. Please include the following information in your emailed intent to quote:

- Company Name
- Address
- Contact Name
- Phone Number
- Email Address

Submittal of your intent to quote in no way obligates your company to submit a quote. However, it will ensure that you will receive any updates that may be made to the RFQ. Failure to email your intent to quote may result in not receiving the updates and, therefore, your response to the RFQ may be deemed unresponsive.

**D. Section Headers**

The section headers and paragraph titles in this RFQ are for informational purposes only and are not legally binding.

**2. ADMINISTRATIVE INFORMATION:**

**A. RFQ Questions and Clarifications**

All inquiries and questions concerning this solicitation must be addressed via email to Kathleen Gallagher at [Kathleen.Gallagher@ogs.ny.gov](mailto:Kathleen.Gallagher@ogs.ny.gov). No inquiries or questions submitted via phone will be accepted. Answers to questions will be provided via addendum. Questions must be submitted by the deadline as stated in Section 2.B - Key Events. When submitting questions to the PRIMARY contact, please include the following in the subject line of your email: “Question for eHBITS RFQ.”

**B. Key Events**

The following key event dates have been established for this project. The most critical date is the final implementation date for the project. OGS seeks an implementation date of November 1, 2012.

RFQ Release Date:	July 12, 2012
Intent to Quote Notice Emailed to the Designated Contact:	July 19, 2012 by 3:00 pm EST
Questions Emailed to the Designated Contact:	July 20, 2012 by 3:00 pm EST
Replies to Questions Emailed to All Bidders (estimated):	July 25, 2012
Complete Proposals Received by the Designated Contact:	August 2, 2012 by 2:00 pm EST
Tentative Bid Winner Selected (estimated):	August 9, 2012
OSC Approval Received (estimated):	September 1, 2012
Contractor Available to Start:	Upon Approval by NYS Office of the State Comptroller
System Go Live:	November 1, 2012

**C. Contract Period**

This contract shall commence after approval by the New York State Office of the State Comptroller and shall be in effect for three (3) months, with an option to extend for two (2) months if the NOT TO EXCEED RATE has not been reached. The contract is expected to begin on or about September 1, 2012.

**D. Price**

Bidders must submit the Quote Proposal Form (located within Appendix B – Required Forms) and include the job title(s) for each person working, the estimated hours for each person proposed, the hourly bill rate for each person proposed, and a NOT TO EXCEED GRAND TOTAL, which shall not exceed \$200,000.

**Bidders must use the job title and skill levels listed in Section D.1 (“eHBITS RFQ Job Titles”) when submitting a response to this RFQ. Bidders who propose candidates using the job titles found on the former IT Services backdrop contract that expired on December 31, 2011, such as Programmer/Analyst I, will result in automatic rejection of the Bidder’s proposal.**

The Hourly Bill Rates shall be deemed INCLUSIVE of travel, meals and lodging. The Contractor agrees that Consultants shall not be separately reimbursed for expenses incurred for travel to and from a designated work location (commuting expenses). The winning vendor will not be able to submit any separate costs other than for the hourly work efforts.

OGS expects that the hourly bill rates proposed shall be competitive and reflect the current marketplace for similar services in New York. The Contractor warrants that pricing offered to OGS is the same as or lower than that offered to its customers who are similarly situated with respect to Consulting Services. In addition, OGS requests that Bidders do not propose candidates where travel expenses will result in significantly increased hourly bill rates.

**1. eHBITS RFQ Job Titles**

<u>Job Title</u>	<u>Definition</u>
Business Analyst	- Manages small to medium-scale business analysis work or projects with distinct deliverables to a solution
Project Manager	- Overseeing projects comprised of multiple deliverables - Delegating and coordinating of tasks - Project status, meetings, scope changes, issues
Tester	- Develops and maintains user and technical documentation and project process documentation for Application Teams - Understand the user’s view of applications and /or technology and are able to put procedures in a logical sequence
Programmer	- Analysis, design, coding, component and assembly testing of all application code owned by the Application Team - Maintenance (including production support), enhancement and development work - Write application software, data analysis, data access, data structures, data manipulation, databases, design, programming, testing and implementation, technical and user documentation, software conversions
Technical Architect	- Technical expert centered around a technology, technologies or a portfolio of

RFQ #1768 – IT Services to Develop a Web-Based Application “eHBITS”

<u>Job Title</u>	<u>Definition</u>
	<p>applications</p> <ul style="list-style-type: none"> <li>- Designing and implementing Information Technology solutions</li> </ul>
Technical Writer	<ul style="list-style-type: none"> <li>- Translates technical information into clear, readable documents to be used by technical and non-technical personnel</li> <li>- Develop and/or maintain the following types of documentation: system documentation; user manuals; installation guides; computer operations and program maintenance manuals; plans for training, testing, quality assurance, and contingency operations; and backup, recovery and restart procedures; technical writing for proposals, presentations, standard operating procedures (SOP), policies and procedures as well as reports</li> <li>- Edits functional descriptions, system specifications, user manuals, special reports, or any other customer deliverables and documents</li> </ul>
Specialist	<ul style="list-style-type: none"> <li>- Experience in the usage and support of a collection of development platforms, technical architectures, or business applications and products that run on those platforms, beyond that of a Programmer</li> <li>- The Authorized User may seek Specialists in the following areas (list is not exhaustive): Backup and Recovery Security</li> </ul>
System Administrator	<ul style="list-style-type: none"> <li>- Server back up and security</li> <li>- Performance tuning and capacity planning</li> <li>- Operations and troubleshooting</li> <li>- Understanding of network and distributed computing concepts</li> </ul>
Database Administrator	<ul style="list-style-type: none"> <li>- Responsible for data analysis and database management</li> <li>- Maintenance, enhancement, designing of data dictionaries, physical and logical database models, and performance tuning</li> </ul>

<u>Skill Level</u>	<u>Typical Experience</u>	<u>Preferred Skills</u>
Junior	12 - 36 Months	Candidate is able to work as a member of the team
Mid Level	36 - 60 Months	Candidate is able to work without assistance
Senior	60 - 84 Months	Candidate is able to work without assistance and can provide leadership for others; may have advanced education
Expert	84+ Months	Candidate is able to provide leadership of large teams and/or extensive industry experience and is considered at the top of his/her field

**E. Submission Documents**

OGS is seeking a detailed Scope of Work (SOW) that should include (but not necessarily be limited to) the following work areas:

- An overview of the work to be performed for OGS;
- The expected responsibilities for OGS;
- Any assumptions of the vendor;
- A resume of each person(s) who will be assigned to work on this project
  - The resume must include at least two (2) professional references from previous assignments.
  - The resume must include projects and accompanying descriptions that are similar in size and scope as this RFQ
- A description of the day to day tasks to be performed by each individual proposed
- A list of the proposed project deliverables.
- All completed forms from Appendix B – Required Forms.

Proposal Security - The content of each Bidder’s proposal will be held in strict confidence during the RFQ evaluation process, and details of a proposal will not be revealed (except as may be required under the Freedom of Information Law or other State Law).

**F. Bid Delivery**

**Bidders assume all risks for timely, properly submitted deliveries.** Bidders are strongly encouraged to arrange for delivery of quotes to OGS **prior to** the date of the Quote opening. **LATE QUOTES will be rejected.** Faxed quote submissions are **NOT** acceptable.

A Bidder **MUST** supply two (2) original signed copies and one (1) electronic version of its proposal. The electronic version shall be submitted on a CD in either PDF or MS Word format.

**Responses to this RFQ should be returned by the deadline, as stated in Section 2.B – Key Events, to:**

**Mail: NYS Office of General Services  
Division of Financial Administration  
Corning Tower - 40<sup>th</sup> Floor Reception Desk  
Empire State Plaza  
Albany, NY 12242  
Attn: Kathleen Gallagher**

**G. Method of Award**

The vendor that represents the best value to the State of New York will be awarded the project. Best Value will be based upon how well the vendor understands the requirements of this project, the experience and skills of the staff to be assigned to work on this project, reference checks of the assigned staff, and the overall cost of the project.

A Service Agreement (an example titled Appendix C is attached) will be forwarded to the vendor awarded this project.

**H. Payment and Final Acceptance (Retainage)**

Invoices shall be submitted for payment at the end of each month for hourly work efforts completed during the month. Each invoice must detail the consultant name, title, hours worked, and the hourly bill rate.

The selected Contractor will be paid for invoices up to 85% of the Not to Exceed Grand Total as bid in the Quote Proposal Form. Upon completion and final acceptance in writing by OGS of all agreed upon deliverables, any remaining billable hours (if any), up to the Not to Exceed Grand Total cost as bid in the Quote Proposal Form, shall be payable.

**All invoices must be submitted for payment to:**

**THE CLAIMS UNIT  
DIVISION OF FINANCIAL ADMINISTRATION -or- [claimsunit@ogs.ny.gov](mailto:claimsunit@ogs.ny.gov)  
OFFICE OF GENERAL SERVICES  
EMPIRE STATE PLAZA STATION  
P. O. BOX 2117  
ALBANY, NEW YORK 12220-0117**

All invoices must be accompanied by signed timesheets or other verification that the hours billed correspond with the hours worked.

**I. Electronic Payments**

Contractor shall provide complete and accurate billing invoices in order to receive payment. Billing invoices submitted must contain all information and supporting documentation required by the contract, the agency, and the State Comptroller. Payment for invoices submitted by the contractor shall only be rendered electronically unless payment by paper check is expressly authorized by the Commissioner, in the Commissioner’s sole discretion, due to extenuating circumstances. Such electronic payment shall be made in accordance with ordinary State procedures and practices. The Contractor shall comply with the State Comptroller’s procedures to authorize electronic payments. Authorization forms are available at the State Comptrollers website at [www.osc.state.ny.us/epay/index.htm](http://www.osc.state.ny.us/epay/index.htm), by e-mail at [epunit@osc.state.ny.us](mailto:epunit@osc.state.ny.us), or by phone at 518-474-4032. Contractor acknowledges that it will not receive payment on any invoices submitted under this Contract if it does not comply with the State Comptroller’s electronic payment procedures, except where the Commissioner has expressly authorized payment by paper check as set forth above.

Please note that in conjunction with New York State’s implementation of a new Statewide financial system, the Office of the State Comptroller requires all vendors doing business with New York State agencies to complete a substitute W-9 form. Vendors registering for electronic payment can complete the W-9 form when they register. Vendors already registered for electronic payment are requested to go to the above website and complete the Substitute W-9 form and submit following the instructions provided.

**3. SCOPE OF WORK:**

**A. In-Scope Work**

New York State Executive Agencies spent approximately \$140 million on HBITS during the 2010-2011 State Fiscal Year. Historically, the amount spent has been distributed across several contracting vehicles and across multiple Executive Agencies with little consistency and continuity. The result was a very inefficient HBITS program and process.

OGS recently conducted a statewide RFP for Hourly Based IT Services (HBITS) to develop a streamlined and cost efficient process to acquire IT consultants. As provided by Exhibit 1 of this RFQ, the candidate selection process and accompanying evaluation forms have been finalized and can be used by NYS governmental entities upon approval of the resultant HBITS contracts. However, OGS is seeking to automate this process by leveraging MS Office products, most notably SharePoint and Infopath, to achieve greater efficiency.

The selected vendor, working with OGS Information Resource Management (IRM) staff, will develop a web portal solution/application with a MS Office solution for the back-end database, as described in Exhibit 2, to be utilized by both OGS and Authorized Users of the HBITS contracts. The MSP will have the ability to see manage the HBITS process through a working MS Office system. IRM will establish a development area in which the vendor will be able to code the application. The responsibility to migrate any part of the application to the test and production platforms will fall to IRM.

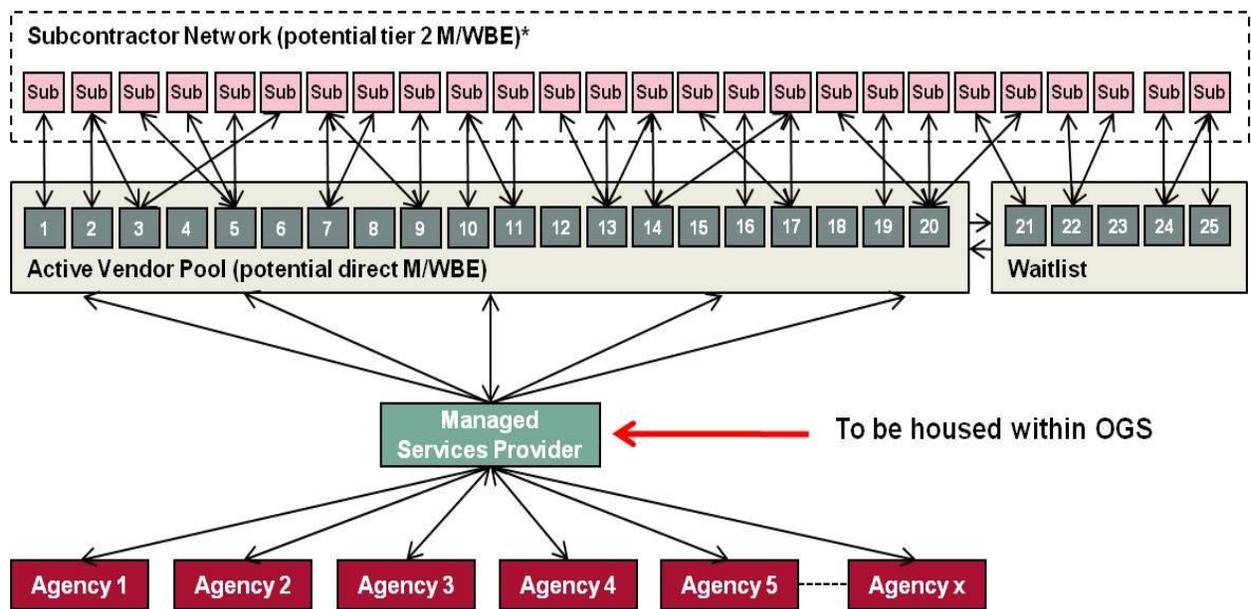
The winning vendor is expected to perform their work at the office of OGS IRM, located on the 36<sup>th</sup> floor of the Corning Tower, Albany, NY.

### 1. Background for eHBITS

The HBITS Contract is not a Continuous Recruitment contract. Rather, this was a competitive RFP for which 25 awards were made. The awardees will comply with a uniform set of Terms and Conditions. OGS will maintain twenty (20) Active Contractors and up to five (5) Waitlisted Contractors at any given point during the HBITS Contract. Contractors will be evaluated on their performance on an annual basis.

As a result of this evaluation, the five (5) Active Contractors with the lowest performance scores will be moved to the Waitlist at the end of each Contract year and the five (5) Waitlisted Contractors will be moved to Active status. This performance cycle will repeat at the end of each 12 month period; a Contractor will not be on the Waitlist for more than one (1) year at a time.

Requests for placements of individual consultants under the HBITS contract will only be distributed to the “Active” Contractors. All Executive Agency placements under the HBITS contract will go through a State-run, centralized organization, housed within OGS, that will handle all administrative aspects of placing consultants, including centralized billing and payment for Executive Agency purchases, as well as several other key activities. This organization will act as a Managed Services Provider (MSP), and will serve as a key point of contact with the Contractor pool. The following sets forth a pictorial representation of how the Executive Agencies, the MSP and the Contractor will exchange information regarding opportunities.



\*Model is illustrative; some vendors have sufficient in-house resources while others will sub-contract business to other vendors

### 2. HBITS Process for Executive Agencies

The process described below is also available in Exhibit 1. The process has been developed with the ability to implement with or without an active secure portal. The steps have been described with either process mediums available, e-mail or portal.

## RFQ #1768 – IT Services to Develop a Web-Based Application “eHBITS”

When a need is identified for hourly-based IT support in an Executive Agency, the Executive Agency Authorized User must have, internal/control approvals **prior** to initiating the eHBITS process. Upon final internal/control agency approval, the Executive Agency Authorized User can notify the MSP of its HBITS need by submitting a completed Task Order Request Form (see Exhibit 1) via email or through the eHBITS portal. The Task Order Request Form will identify the selected Job Titles needed, as well as other requested qualifications to meet agency need.

Once the MSP receives the Task Order Request Form, it is reviewed and verified for completeness. The MSP will then distribute an email, with the Task Order Request Form attached, to all Active Contractors and/or post to the Contractor bulletin board, via the portal within three business days. At this time, contractors will review the Request Form, and complete a Candidate Response Form (Exhibit 1) to apply for the posted job. Circumstances may arise in which an Executive Agency Authorized User has an urgent HBITS need. In this case the Executive Agency Authorized User will indicate on the Task Order Request Form that a five (5) business day turnaround time is required for Contractors to return Candidate Response Forms. This will occur in limited circumstances and is subject to written pre-approval by the MSP. The failure of an Executive Agency Authorized User to plan for expected gaps in HBITS services shall not be considered as an adequate justification to receive approval for the five (5) business day turnaround time.

The Executive Agency Authorized User can request a minimum of one (1) and a maximum of (2) Candidate Response Forms per position from each Active Contractor. The MSP will accept responses to the Task Order Request Form from Active Contractors through 5:00 PM EST on the tenth Business Day (exclusive of the day of transmission to the Contractors), unless it is titled as an “expedited” request through 5:00 PM on the fifth Business Day. Any requests for clarification on a Task Order Request Form must be submitted in writing to the MSP.

Once Candidate Response Forms have been collected or received through the secured portal from the contractors, the MSP will review the proposed Hourly Bill Rates and ensure proposed qualifications are in accordance with the contract. Hourly Bill Rates must satisfy the requirements of State Finance Law § 163(10) (c). In addition, the

MSP will review the Candidate Response Forms to ensure that the proposed Candidates meet the mandatory qualifications. Executive Agency Authorized Users are not allowed to add additional “mandatory” qualifications to the Task Order Request Form requested job title and skill level. Candidate Response Forms that do not meet mandatory position requirements shall be rejected.

Finally, the MSP shall use the following methodology to determine the total number of qualified Candidate Response Forms to be passed on to the Executive Agency Authorized User for each position sought:

- a. If 10-40 qualified Candidates: 50% of the Candidate Response Forms with the lowest Hourly Bill Rates are forwarded to the Executive Agency Authorized User. If the 11<sup>th</sup> lowest Hourly Bill Rate is within 1% of the 10<sup>th</sup> lowest Hourly Bill Rate, that Candidate Response Form would also be forwarded to the Executive Agency Authorized User.
- b. If 6-10 qualified Candidates: Candidate Response Forms with 5 lowest Hourly Bill Rates are forwarded to the Executive Agency Authorized User. If the 6<sup>th</sup> lowest Hourly Bill Rate is within 1% of the 5<sup>th</sup> lowest Hourly Bill Rate, that Candidate Response Form would also be forwarded to the Executive Agency Authorized User.
- c. If 5 or fewer qualified Candidates: MSP will investigate and determine whether to restart the process or forward all Candidate Response Forms.

The MSP may elect to perform additional screening of Candidate Response Forms. Upon completion of the screening, suitable Candidate Response Forms (up to 40) are passed on to the Executive Agency Authorized User by 5:00 PM EST no later than on the second business day after the close of the submission period from Contractors (exclusive of receipt date), via e-mail or through the portal.

## RFQ #1768 – IT Services to Develop a Web-Based Application “eHBITS”

Upon receipt on the Candidate Response Forms from the MSP, the Executive Agency Authorized User must make every attempt to evaluate the Candidate Response Forms within five (5) business days of receipt and notify the Contractors via an e-mail to or through the MSP of the Candidates selected for interview. The Executive Agency Authorized User shall interview at least the minimum number of Candidates for the number of positions sought in accordance with the below table. An Executive Agency Authorized User is required to interview the three (3) Candidates with the highest score resulting from the preliminary technical evaluation before it may select a lower ranked Candidate for the position.

<b>Number of Positions Sought through a Single Task Order</b>	<b>Minimum Number of Candidate Interviews</b>
1	3
2	5
3	6
4	8
5	10

By the end of the five (5) business days after receipt of the Response Forms, the Executive Agency Authorized User will then notify the MSP of the preliminary technical evaluation results via Form 3A (Exhibit 1) via e-mail or portal. The MSP will then inform the Contractors of the interview results. Proposed candidates will either be scheduled for an interview or are released for other jobs. In the event that two Active Contractors submit the same Candidate for the same request, only the Contractor with the lower Hourly Bill Rate shall be considered. In the event a Candidate is submitted for different requests with overlapping schedules, only the submission to the earliest announced request shall be considered. In neither case will the Contractor that submitted the rejected Candidate Response Form be permitted to resubmit a new Candidate.

The MSP will not schedule interviews on behalf of the Executive Agency Authorized User. If the Executive Agency Authorized User does not notify the MSP of the preliminary technical evaluation results within five (5) business days, the Candidates shall be deemed released. Once a Candidate is released, the Contractor may resume submitting the proposed staff for other requests. Candidates selected for interview will not be considered released until the final selection is made by the Authorized User.

Once the Authorized User completes interviewing, scores and results are posted in the Evaluation Form (Form 3B – Exhibit 1). The AU updates the MPS with the final section via the submission for Form 3B on e-mail or through the portal. At this time, the non-selected candidates are released and available to be proposed on other Task Orders.

The Executive Agency Authorized User reserves the right to reject all Candidate Response Forms provided by the Active Contractors. In this case, the MSP may re-post the Task Order Request Form to the Active Contractor pool for Candidate Response Forms. Upon submission for a specified Task Order Request Form, a Contractor cannot submit the same Candidate for another position until such Candidate is released from consideration by the Authorized User.

Upon final selection, Authorized User begins the on-boarding process, and the Contractor has 10 business days to perform all Candidate placement requirements and finalizes all on-boarding requirements by the Authorized User. Throughout the eHBITS process, the MSP will have the ability to manage, monitor and record data through a working Microsoft Office database.

### 3. HBITS Process for Non-MSP (Non-Executive Agency) Users

**The process described below is also available in Exhibit 1. The process has been developed with the ability to implement with or without an active secure portal. The steps have been described with either process mediums available, e-mail or portal.**

When a need is identified for hourly-based IT support in a Non-Executive Agency, the Authorized User is required to have all internal/control agency approvals necessary **prior** to initiating the HBITS process. Upon internal/control agency approvals, the Non-Executive Agency Authorized User will complete a Task Order Request Form and send it to the MSP for dissemination via email or through the eHBITS portal. The Task Order Request Form will identify the selected Job Titles needed, as well as additional requested qualifications as well as other requested qualifications to meet agency need. The Non-Executive Agency Authorized User does **not** send the Task Order Request Form directly to the Active Contractors.

The MSP will distribute an e-mail to all Active Contractors and/or post to the Contractor bulletin board the completed Task Order Request Forms. However, the Non-Executive Agency Authorized User will accept Candidate Response Forms, instead of the MSP, directly from the Active Contractors through 5:00PM EST on the tenth Business Day (exclusive of the day of transmission to the Contractors). **Non-Executive Agency Authorized Users shall not be allowed to request a five (5) business day turnaround time for Contractors to propose Candidates.** Any requests for clarification on a Task Order Request Form must be submitted in writing to the agency contact name identified in the Task Order Request Form.

The remainder of the candidate selection process is the same as Executive Agencies.

### 4. Portal Solution

To manage the HBITS contract, OGS seeks the development of a web-portal solution to be known as “eHBITS” to manage the HBITS candidate selection process, as described above and depicted in Exhibit 1. The proposed solution will facilitate the following:

- The ability to post task order requests by Authorized Users through a secure portal between AUs and Contractors
- The ability to send E-mail notifications to Active Contractors and AUs
- The ability to facilitate the Candidate selection process
- The ability to create a consistent, qualified and fair process for HBITS users
- The ability of AUs and Contractors to have easy access to a working portal from the main OGS website
- The ability of the MSP to manage, monitor and record eHBITS data through a MS Office database (back-end)

Exhibit 2 is a “working” PowerPoint document developed to visually represent proposed portal. Exhibit 2 is for information only. Changes and modifications are allowed. The web-portal solution proposed and/or developed by the selected Bidder must incorporate a MS Office product, such as SharePoint and InfoPath.

### 5. OGS Responsibilities

OGS will assign a project manager ("Project Manager") to:

- Provide direction and guidance to OGS personnel as required by the Contractor to maintain project momentum.
- Provide information and resources in a timely manner as needed by Contractor to enable the Contractor to complete the tasks described in this SOW.
- Be readily available when required by Contractor for the duration of the Contract.
- The Project Manager will be responsible for receiving and approving any deliverables created as a result of this Service.

OGS will provide Contractor with the following:

- Adequate workspace for each of its consultants, as well as access to telephones, copiers, faxes, conference rooms, and printing facilities as reasonably necessary.
- Access to key OGS personnel, including business, IT and operational staff.
- Copies of relevant configuration and processes documentation.
- Facilities access and access to relevant internal and external systems as needed.

## **6. Contractor Assumptions and Dependencies**

Contractor will rely on the following assumptions and dependencies, together with those stated elsewhere in this RFQ, in performing the Service

- Contractor will coordinate project management activities with an OGS Project Manager.
- Contractor will have primary responsibility for coordinating all activities for this Service, including scheduling resources, and confirming project activities and deliverables are within the scope of this RFQ.
- Contractor shall identify a single point of contact for this Contract.
- Meetings will take place at an OGS Facility. However, Contractor can make its facilities available if required.
- Any Service schedule estimates represent Contractor's best technical judgment based on information available.
- Contractor may not use subcontractors to perform any work, unless prior written consent is received from OGS.
- Unless specifically stated otherwise, Contractor will provide all deliverables associated with this RFQ on a time and-materials basis.
- Contractor will propose candidates who have excellent communication skills including the ability to clearly communicate in English.

### **B. Confidentiality**

Contractor agrees to keep confidential and not to disclose to third parties any information provided by the OGS or learned by the Contractor during the performance of the Contract unless Contractor has received the prior written consent of the OGS to make such disclosure. This provision shall survive the expiration and termination of this Contract. The Contractor warrants that all of its operations are compliant with all federal, state and local laws, rules and regulations pertain to the privacy and/or security of personal and confidential information.

Contractor agrees to sign a non-disclosure agreement to be provided by OGS at the start of the agreement.

### **C. Out-of-Scope Work**

The vendor selected as a result of the HBITS RFQ:

- Will not act as the Managed Services Provider for the HBITS contract.
- Will not be an approved vendor to provide Hourly-Based IT Services (HBITS) under the HBITS contracts
- Will not retain ownership nor proprietary rights to the solution developed under the contract resulting from this RFQ.

In addition, this RFQ is not a backdrop contract nor a replacement for the IT Services backdrop contract that expired on December 31, 2011.

**4. CONTRACT CLAUSES AND REQUIREMENTS:**

**A. Appendix A**

Appendix A, Standard Clauses For New York State Contracts, dated December 2011, attached hereto, is hereby expressly made a part of this Request for Quote Document as fully as if set forth at length herein. **Please retain this document for future reference.**

**B. Conflict of Terms and Conditions**

Conflicts between documents shall be resolved in the following order of precedence:

- a. Appendix A
- b. Request for Quote 1768 (this document) with any addendum(a)
- c. Bidder’s Quote

**C. Summary of Policy and Prohibitions on Procurement Lobbying**

Pursuant to State Finance Law §§139-j and 139-k, this solicitation includes and imposes certain restrictions on communications between OGS and an Offerer/bidder during the procurement process. An Offerer/bidder is restricted from making contacts from the earliest notice of intent to solicit offers/bids through final award and approval of the Procurement Contract by OGS and, if applicable, the Office of the State Comptroller (“restricted period”) to other than designated staff unless it is a contact that is included among certain statutory exceptions set forth in State Finance Law §139-j (3) (a). Designated staff, as of the date hereof, is identified on the first page of this solicitation. OGS employees are also required to obtain certain information when contacted during the restricted period and make a determination of the responsibility of the Offerer/bidder pursuant to these two statutes. Certain findings of non-responsibility can result in rejection for contract award and in the event of two findings within a four-year period the Offerer/bidder is debarred from obtaining governmental Procurement Contracts. Further information about these requirements can be found on the OGS website:

<http://www.ogs.state.ny.us/aboutOgs/regulations/defaultAdvisoryCouncil.html>

**D. Contractor Insurance Requirements**

Prior to the commencement of the work to be performed by the Contractor hereunder, the Contractor shall file with The People of the State of New York, Office of General Services (hereinafter referred to as “OGS”), Certificates of Insurance (hereinafter referred to as “Certificates”), evidencing compliance with all requirements contained in this Contract. Such Certificates shall be of a form and substance acceptable to OGS.

Certificate acceptance and/or approval by OGS does not and shall not be construed to relieve Contractor of any obligations, responsibilities or liabilities under the Contract.

The Contractor, throughout the term of this Contract, or as otherwise required by this Contract, shall obtain and maintain in full force and effect, the following insurance with limits not less than those described below and as required by the terms of this Contract, or as required by law, whichever is greater (limits may be provided through a combination of primary and umbrella/excess policies):

**WORKERS’ COMPENSATION / DISABILITY INSURANCE:**

Workers’ Compensation, Employer’s Liability, and Disability Benefits meeting all New York State statutory requirements are required. If coverage is obtained from an insurance company through an insurance policy, the policy shall provide coverage for all states of operation that apply to the performance of the contract. In addition, if employees will be working on, near or over navigable waters, coverage provided under the US Longshore and Harbor Workers’ Compensation Act must be included. Also, if the contract is for temporary services, or involves renting equipment with operators, the Alternate Employer Endorsement, WC 00 03 01A, must be included on the policy naming the People of the State of New York as the alternate employer.

RFQ #1768 – IT Services to Develop a Web-Based Application “eHBITS”

**PROOF of COMPLIANCE WITH WORKERS’ COMPENSATION COVERAGE REQUIREMENTS:**

ACORD forms are NOT acceptable proof of workers’ compensation coverage.

In order to provide proof of compliance with the requirements of the Workers’ Compensation Law pertaining to workers’ compensation coverage, contractors shall:

- A) Be legally exempt from obtaining workers’ compensation insurance coverage;  
Or
- B) Obtain such coverage from insurance carriers;  
Or
- C) Be a Board-approved self-insured employer or participate in an authorized self-insurance plan.

Contractors seeking to enter into contracts with the State of New York shall provide one of the following forms to the Office of General Services at the time of bid submission or shortly after the opening of bids:

A) Form CE-200, Certificate of Attestation for New York Entities With No Employees and Certain Out of State Entities, That New York State Workers’ Compensation and/or Disability Benefits Insurance Coverage is Not Required which is available on the Workers’ Compensation Board’s website ([www.wcb.state.ny.us](http://www.wcb.state.ny.us));

Or

B) Certificate of Workers’ Compensation Insurance:

1) Form C-105.2 (9/07) if coverage is provided by the contractor’s insurance carrier, contractor must request its carrier to send this form to the New York State Office of General Services;

Or

2) Form U-26.3 if coverage is provided by the State Insurance Fund, contractor must request that the State Insurance Fund send this form to the New York State Office of General Services;

Or

C) Certificate of Workers’ Compensation Self-Insurance - Form SI-12, available from the New York State Workers’ Compensation Board’s Self-Insurance Office;

Or

D) Certificate of Participation in Workers’ Compensation Group Self-Insurance Form GSI-105.2, available from the contractor’s Group Self-Insurance Administrator.

**PROOF of COMPLIANCE WITH DISABILITY BENEFITS COVERAGE REQUIREMENTS:**

In order to provide proof of compliance with the requirements of the Workers’ Compensation Law pertaining to disability benefits, contractors shall:

- A) Be legally exempt from obtaining disability benefits coverage;  
Or
- B) Obtain such coverage from insurance carriers;  
Or
- C) Be a Board-approved self-insured employer.

Contractors seeking to enter into contracts with the State of New York shall provide one of the following forms to the Office of General Services at the time of bid submission or shortly after the opening of bids:

A) Form CE-200, Certificate of Attestation for New York Entities With No Employees and Certain Out of State Entities, That New York State Workers’ Compensation and/or Disability Benefits Insurance Coverage is Not Required which is available on the Workers’ Compensation Board’s website ([www.wcb.state.ny.us](http://www.wcb.state.ny.us));

Or

B) Form DB-120.1, Certificate of Disability Benefits Insurance. Contractor must request its business insurance carrier to send this form to the New York State Office of General Services;

Or

C) Form DB-155, Certificate of Disability Benefits Self-Insurance. The Contractor must call the Board’s Self-Insurance Office at 518-402-0247 to obtain this form.

**All forms must name the Office of General Services – Financial Administration, 40th Floor, Mayor Erastus Corning 2nd Tower, Empire State Plaza, Albany NY 12242, as the Entity Requesting Proof of Coverage (Entity being listed as the Certificate Holder).**

Waiver of Subrogation. Contractor shall cause to be included in each of its policies a waiver of the insurer’s right of subrogation against OGS, or, if such waiver is unobtainable (i) an express agreement that such policy shall not be invalidated if Contractor waives or has waived before the casualty, the right of recovery against OGS or (ii) any other form of permission for the release of OGS.

Contractor acknowledges that failure to obtain any or all required insurance on behalf of OGS constitutes a material breach of contract and subjects it to liability for damages, indemnification and all other legal remedies available to OGS.

**E. Employee Information to be Reported by Certain Consultant Contractors**

Chapter 10 of the Laws of 2006 amended the Civil Service Law and the State Finance Law, relative to maintaining certain information concerning contract employees working under State agency service and consulting contracts. State agency consultant contracts are defined as “contracts entered into by a state agency for *analysis, evaluation, research, training, data processing, computer programming, engineering, environmental health and mental health services, accounting, auditing, paralegal, legal, or similar services*” (“covered consultant contract” or “covered consultant services”). The amendments also require that certain contract employee information be provided to the state agency awarding such contracts, the Office of the State Comptroller (OSC), the Division of the Budget and the Department of Civil Service (CS). The effective date of these amendments is June 19, 2006. The requirements will apply to covered contracts awarded on and after such date.

To meet these new requirements, the Contractor agrees to complete:

Form A - the Contractor’s Planned Employment Form upon bid/quote submittal.

Form B - the Contractor’s Annual Employment Report throughout the term of the Contract by May 1<sup>st</sup> of each year. The following information must be reported:

**For each covered consultant contract in effect at any time between the preceding April 1<sup>st</sup> through March 31<sup>st</sup> fiscal year or for the period of time such contract was in effect during such prior State fiscal year:**

- 1. Total number of employees employed to provide the consultant services, by employment category.**
- 2. Total number of hours worked by such employees.**
- 3. Total compensation paid to all employees that performed consultant services under such Contract.\***

(Information must be reported on the Contractor’s Annual Employment Report (Form B) or other format stipulated by OGS.)

**\*NOTE: The information to be reported is applicable only to those employees who are directly providing services or directly performing covered consultant services. However, such information shall also be provided relative to employees of Subcontractors who perform any part of the service contract or any part of the covered consultant contract. This information does not have to be collected and reported in circumstances where there is ancillary involvement of an employee in a clerical, support, organizational or other administrative capacity.**

**F. Tax and Finance Clause**

**TAX LAW § 5-A:**

Section 5-a of the Tax Law, as amended, effective April 26, 2006, requires certain contractors awarded state contracts for commodities, services and technology valued at more than \$100,000 to certify to the Department of Taxation and Finance (DTF) that they are registered to collect New York State and local sales and compensating use taxes. The law applies to contracts where the total amount of such contractors’ sales delivered into New York State are in excess of \$300,000 for the four quarterly periods immediately preceding the quarterly period in which the certification is made, and with respect to any affiliates and subcontractors whose sales delivered into New York State exceeded \$300,000 for the four quarterly periods immediately preceding the quarterly period in which the certification is made.

This law imposes upon certain contractors the obligation to certify whether or not the contractor, its affiliates, and its subcontractors are required to register to collect state sales and compensating use tax and contractors must certify to DTF that each affiliate and subcontractor exceeding such sales threshold is registered with DTF to collect New York State and

local sales and compensating use taxes. The law prohibits the State Comptroller, or other approving agency, from approving a contract awarded to a contractor meeting the registration requirements but who is not so registered in accordance with the law.

Contractor certification forms and instructions for completing the forms are attached to this IFB. Form ST-220-TD must be filed with and returned directly to DTF. Unless the information upon which the ST-220-TD is based changes, this form only needs to be filed once with DTF. If the information changes for the contractor, its affiliate(s), or its subcontractor(s) a new Form ST-220-TD must be filed with DTF.

Form ST-220-CA must be filed with the bid and submitted to the procuring covered agency certifying that the contractor filed the ST-220-TD with DTF. Proposed contractors should complete and return the certification forms within two business days of request (if the forms are not completed and returned with bid submission). Failure to make either of these filings may render a Bidder non-responsive and non-responsible. Bidders shall take the necessary steps to provide properly certified forms within a timely manner to ensure compliance with the law.

Vendors may call DTF at **1-800-698--2909** for any and all questions relating to Section 5-(a) of the Tax Law and relating to a company’s registration status with the DTF. For additional information and frequently asked questions, please refer to the DTF web site: <http://www.nystax.gov>.

**G. Contractor Requirements and Procedures for Business Participation Opportunities for New York State Certified Minority and Women-Owned Business Enterprises and Equal Employment Opportunities for Group Members and Women**

**NEW YORK STATE LAW**

Pursuant to New York State Executive Law Article 15-A, OGS recognizes its obligation under the law to promote opportunities for maximum feasible participation of certified minority and women-owned business enterprises and the employment of minority group members and women in the performance of OGS contracts.

In 2006, the State of New York commissioned a disparity study to evaluate whether minority and women-owned business enterprises had a full and fair opportunity to participate in state contracting. The findings of the study were published on April 29, 2010, under the title "The State of Minority and Women-Owned Business Enterprises: Evidence from New York" (“Disparity Study”). The report found evidence of statistically significant disparities between the level of participation of minority and women-owned business enterprises in state procurement contracting versus the number of minority and women-owned business enterprises that were ready, willing and able to participate in state procurements. As a result of these findings, the Disparity Study made recommendations concerning the implementation and operation of the statewide certified minority and women-owned business enterprises program. The recommendations from the Disparity

Study culminated in the enactment and the implementation of New York State Executive Law Article 15-A, which requires, among other things, that OGS establishes goals for maximum feasible participation of New York State Certified minority and women-owned business enterprises (“MWBE”) and the employment of minority groups members and women in the performance of New York State contracts.

### **Equal Employment Opportunity Requirements**

By submission of a bid or proposal in response to this solicitation, the Bidder/Contractor agrees with all of the terms and conditions of Appendix A including Clause 12 - Equal Employment Opportunities for Minorities and Women. The Contractor is required to ensure that it and any subcontractors awarded a subcontract over \$25,000 for the construction, demolition, replacement, major repair, renovation, planning or design of real property and improvements thereon (the "Work") except where the Work is for the beneficial use of the Contractor, shall undertake or continue programs to ensure that minority group members and women are afforded equal employment opportunities without discrimination because of race, creed, color, national origin, sex, age, disability or marital status. For these purposes, equal opportunity shall apply

in the areas of recruitment, employment, job assignment, promotion, upgrading, demotion, transfer, layoff, termination, and rates of pay or other forms of compensation. This requirement does not apply to: (i) work, goods, or services unrelated to the Contract; or (ii) employment outside New York State.

Bidder further agrees to submit with the bid a staffing plan on Form EEO 100 identifying the anticipated work force to be utilized on the Contract and if awarded a Contract, will, upon request, submit to OGS, a workforce utilization report identifying the workforce actually utilized on the Contract if known.

Further, pursuant to Article 15 of the Executive Law (the “Human Rights Law”), all other State and Federal statutory and constitutional non-discrimination provisions, the Contractor and sub-contractors will not discriminate against any employee or applicant for employment because of race, creed (religion), color, sex, national origin, sexual orientation, military status, age, disability, predisposing genetic characteristic, marital status or domestic violence victim status, and shall also follow the requirements of the Human Rights Law with regard to non-discrimination on the basis of prior criminal conviction and prior arrest.

### **Business Participation Opportunities for MWBEs**

For purposes of this procurement, OGS has conducted a comprehensive search and has determined that the contract does not offer sufficient opportunities to set goals for participation by MWBEs as subcontractors, service providers and suppliers to the awarded contractors.

**Please Note: Failure to comply with the foregoing requirements may result in a finding of non-responsiveness, non-responsibility and/or a breach of the Contract, leading to the withholding of funds, suspension or termination of the Contract or such other actions or enforcement proceedings as allowed by the Contract.**

**ALL FORMS ARE AVAILABLE AT: <http://www.ogs.ny.gov/MWBE/Forms.asp>**

### **H. New York State Vendor File Registration**

Prior to being awarded a Contract pursuant to this Solicitation, the Bidder(s) and any designated authorized reseller(s) who accept payment directly from the State must be registered in the New York State Vendor File (Vendor File) administered by the OSC. This is a central registry for all vendors who do business with New York State Agencies and the registration must be initiated by a State Agency. Following the initial registration, a unique New York State ten-digit vendor identification number (Vendor ID) will be assigned to your company and Vendor IDs will be assigned to each of your authorized resellers (if any) for usage on all future transactions with New York State. Additionally, the Vendor File enables vendors to use the Vendor Self-Service application to manage certain vendor information in one central location for all transactions related to the State of New York.

**If the Bidder is already registered in the Vendor File**, the vendor must enter the vendor’s ten-digit Vendor ID on the first page of this bid document. An authorized reseller already registered in the Vendor File must enter its ten-digit Vendor ID along with the authorized reseller’s information on the first page of this bid document.

**If the Bidder is not currently registered in the Vendor File**, the bidder must request assignment of a Vendor ID number from OGS. Complete the OSC Substitute W-9 Form ([http://www.osc.state.ny.us/vendors/forms/ac3237\\_fe.pdf](http://www.osc.state.ny.us/vendors/forms/ac3237_fe.pdf)) and submit the form to OGS **in advance** of your bid. Please send this document to the Designated Contact in the solicitation. In addition, if an authorized reseller(s) is to be used that does not have a Vendor ID, an OSC Substitute W-9 form ([http://www.osc.state.ny.us/vendors/forms/ac3237\\_fe.pdf](http://www.osc.state.ny.us/vendors/forms/ac3237_fe.pdf)) should be completed by each designated authorized reseller and submitted to OGS. **The OGS will initiate the vendor registration process** for all Bidders and their authorized resellers. Once the process is initiated, registrants will receive an e-mail identifying their unique ten-digit Vendor ID and instructions on how to enroll in the online Vendor Self-Service application. For more information on the Vendor File please visit the following website: [http://www.osc.state.ny.us/vendor\\_management/](http://www.osc.state.ny.us/vendor_management/).

#### **I. NYS Standard Vendor Responsibility Questionnaire**

OGS conducts a review of prospective contractors (“Bidders”) to provide reasonable assurances that the Bidder is responsive and responsible. A Questionnaire is used for non-construction contracts and is designed to provide information to assess a Bidder’s responsibility to conduct business in New York based upon financial and organizational capacity, legal authority, business integrity, and past performance history. By submitting a bid, Bidder agrees to fully and accurately complete the “Questionnaire.” The Bidder acknowledges that the State’s execution of the Contract will be contingent upon the State’s determination that the Bidder is responsible, and that the State will be relying upon the Bidder’s responses to the Questionnaire when making its responsibility determination.

OGS recommends each Bidder file the required Questionnaire online via the New York State VendRep System. To enroll in and use the VendRep System, please refer to the VendRep System Instructions and User Support for Vendors available at the Office of the State Comptroller’s (OSC) website, [http://www.osc.state.ny.us./vendrep/vendor\\_index.htm](http://www.osc.state.ny.us./vendrep/vendor_index.htm) or to enroll, go directly to the VendRep System online at <https://portal.osc.state.ny.us>.

OSC provides direct support for the VendRep System through user assistance, documents, online help, and a help desk. The OSC Help Desk contact information is located at <http://www.osc.state.ny.us/portal/contactbuss.htm>. Bidders opting to complete the paper questionnaire can access this form and associated definitions via the OSC website at: [http://www.osc.state.ny.us/vendrep/forms\\_vendor.htm](http://www.osc.state.ny.us/vendrep/forms_vendor.htm).

**In order to assist the State in determining the responsibility of the Bidder, the Bidder should complete and certify (or recertify) the Questionnaire no more than six (6) months prior to the RFQ due date.**

A Bidder’s Questionnaire cannot be viewed by OGS until the Bidder has certified the Questionnaire. It is recommended that all Bidders become familiar with all of the requirements of the Questionnaire in advance of the bid opening to provide sufficient time to complete the Questionnaire.

The Bidder agrees that if it is found by the State that the Bidder’s responses to the Questionnaire were intentionally false or intentionally incomplete, on such finding, OGS may terminate the Contract. In no case shall such termination of the Contract by the State be deemed a breach thereof, nor shall the State be liable for any damages for lost profits or otherwise, which may be sustained by the Contractor as a result of such termination.

#### **J. General Requirements**

1. For discrepancies between the RFQ and the Bid involving the nature, quality, or scope of services to be furnished, it shall be assumed that the Contractor has based the bid on the more expensive manner. Final decision will rest with the Commissioner.
2. The State shall not be liable for any expense incurred by the Contractor as a consequence of any traffic infraction or parking violations attributable to employees of the Contractor.
3. OGS’s interpretation of specifications shall be final and binding upon the Contractor.

**K. Dispute Resolution**

For purposes of this RFQ it is the intention of the Office of General Services’ Financial Administration (OGS) to provide Proposers with an opportunity to administratively resolve disputes, complaints or inquiries related to proposals, this solicitation or contract awards. OGS encourage vendors to seek resolution of disputes through consultation with OGS Financial Administration staff. All such matters will be accorded impartial and timely consideration. Interested parties may also file formal written disputes. A copy of the OGS Financial Administration Dispute Resolution Procedures for Vendors may be obtained by contacting the designated contact person identified on the front of the solicitation document.

**L. Termination**

1. Standard Termination

The Office of General Services may, upon twenty (20) calendar days notice, terminate the contract resulting from this RFQ in the event of the awarded Bidder’s failure to comply with any of the proposal’s requirements unless the awarded Bidder obtained a waiver of the requirement.

In addition, OGS may also terminate any contract resulting from this RFQ upon ten (10) days written notice if the Contractor makes any arrangement or assignment for the benefit of the creditors.

Furthermore, OGS shall have the right, in its sole discretion, at any time to terminate a contract resulting from this RFP/IFB, or any unit portion thereof, with or without cause, by giving thirty (30) days written notice of termination to the Contractor.

Any termination by OGS under this Section shall in no event constitute or be deemed a breach of any contract resulting from this RFQ and no liability shall be incurred by or arise against the Office of General Services, its agents and employees therefore for lost profits or any other damages.

2. Procurement Lobbying Termination

The Office of General Services reserves the right to terminate this Agreement in the event it is found that the certification filed by the Contractor in accordance with New York State Finance Law §139-k was intentionally false or intentionally incomplete. Upon such finding, the Office of General Services may exercise its termination right by providing written notification to the Contractor in accordance with the written notification terms of this Agreement.

**M. Services Warranty**

Contractor shall warrant that the services acquired under this Contract will be provided in a professional manner in accordance with industry standards. OGS must notify Contractor of any services warranty deficiencies within ninety (90) calendar days from performance of the services that gave rise to the warranty claim.

**N. Information Security Breach and Notification Act**

Section 208 of the State Technology Law (STL) and Section 899-aa of the General Business Law (GBL) require that State entities and persons or businesses conducting business in New York who own or license computerized data which includes private information including an individual's unencrypted personal information plus one or more of the following: social security number, driver's license number or non-driver ID, account number, credit or debit card number plus security code, access code or password which permits access to an individual's financial account, must disclose to a New York resident when their private information was, or is reasonably believed to have been, acquired by a person without valid authorization. Disclosure of breach of that private information to all individuals affected or potentially affected must occur in the most expedient time possible without unreasonable delay, after necessary measures to determine the scope of the breach and to restore integrity, but with delay if law enforcement determines it impedes a criminal investigation. When notification is necessary, the State entity or person or business conducting business in New

## RFQ #1768 – IT Services to Develop a Web-Based Application “eHBITS”

York must also notify the following New York State agencies: the Attorney General, the Office of Cyber Security & Critical Infrastructure Coordination (CSCIC) and the Consumer Protection Board (CPB). Information relative to the law and the notification process is available at: <http://www.cscic.state.ny.us/security/securitybreach/>

### **O. Work in the United States**

All work done by Consultants under any Agreement resulting from this RFQ must be performed within the United States.

### **P. Additional Payment Provision**

The State shall not be liable for the payment of any taxes under this Contract however designated, levied or imposed. No person, firm, or corporation is exempt from paying the State Truck Mileage and Unemployment Insurances Taxes and other Federal, State, and local taxes to which the Contractor is subject.

Contractor agrees that its Consultants are engaged to perform services and that OGS shall have full and complete ownership of all deliverables prepared by such Consultant. At the end of the engagement, Contractor agrees that the Consultant shall provide all deliverables and materials to OGS.

### **Q. Use Restrictions and Intellectual Property**

This Statement of Work clarifies that all work by the contractor for OGS is intended as work for hire. The parties agree that the Contractor work is specially ordered and commissioned for use as contributions to a collective work, or is other such work as specified by Section 101(2) of the U.S. Copyright Act [17 U.S.C. 101(2)], and is intended to be a work for hire that is made for the use and ownership of the State of New York and the Office of General Services. Furthermore, the State of New York and Contractor agree that the State of New York is the owner of all copyrights regarding the work. Contractor warrants to the State of New York and the Office of General Services that all of its subcontractors and their employees, who have been, or may be used in regard to this Statement of Work, forfeit all past or future claims of title or ownership to the work produced.

### **R. Consultant Replacements**

OGS recognizes that circumstances may arise that necessitate Consultants to be substituted during the engagement. Replacement of Consultants will not be grounds for an increase in the Hourly Bill Rate, or an extension of the time for completion of the engagement. When providing a replacement Candidate, Contractor must respond in the format of the requested by the RFQ. Any replacement Candidate must meet or exceed all requirements as originally proposed by the Contractor in RFQ #1768.

The Contractor expressly acknowledges that if the Contractor or the chosen Consultant does not fulfill the obligations of the Contract, costs to the Authorized User to replace the Consultant’s services will result, and establishing the precise value of such costs would be difficult and time consuming.

In the event it becomes necessary to replace a Consultant during the engagement, the Contractor shall provide OGS with five (5) business days’ prior written notification describing the circumstances of the need for replacement. The Contractor shall identify a comparable replacement for OGS within five (5) business days. OGS reserves the right to do one of the following:

- Allow the originally selected Contractor to provide a replacement Consultant if the replacement is necessitated by Consultant termination, sickness, or other similar material change in the employment circumstance of the Consultant. Contractor acknowledges that the failure to provide a consultant for the duration of the engagement constitutes a breach of contract and that as a liquidated damage, OGS has the right to receive up to two working weeks (80 hours) of work from the replacement Consultant, at no cost to OGS, during a transition/ramp-up period. This liquidated damage may be waived in whole or in part if it is determined that the need to replace the Consultant was beyond the control of the Contractor.
- Terminate the engagement with the Contractor.

**S. New York State Procurement Rights**

A Bidder is hereby notified that New York State reserves the right to:

1. Reject any or all proposals received in response to the RFQ.
2. Withdraw the RFQ at any time, at the sole discretion of OGS.
3. Make an award under the RFQ in whole or in part.
4. Disqualify any Bidder whose conduct and/or proposal fails to conform to the requirements of the RFQ.
5. Seek clarifications and revisions of proposals.
6. Prior to the bid opening, amend the RFQ specifications to correct errors or oversights, or to supply additional information, as it becomes available.
7. Prior to the bid opening, direct Bidders to submit proposal modifications addressing subsequent RFQ amendments.
8. Eliminate any mandatory, non-material specifications that cannot be complied with by all of the Bidders.
9. Waive any requirements that are not material.
10. Utilize any and all ideas submitted in the bids received.
11. Adopt all or any part of a Bidder's proposal in selecting the optimum configuration.
12. Negotiate with the Bidder responding to this RFQ within the RFQ requirements to serve the best interests of the State. This includes requesting clarifications of any or all Bidders' proposals.
13. Require clarification at any time during the procurement process and/or require correction of arithmetic or other apparent errors for the purpose of assuring a full and complete understanding of a Bidder's proposal and/or to determine a Bidder's compliance with the requirements of the solicitation.
14. Select and award the Contract to other than the selected Bidder in the event of unsuccessful negotiations or, optionally, in other specified circumstances as detailed in the RFQ requirements.
15. Request current Bidder financial statements documenting past sales history that demonstrates ability to service a contract with dollar sales volume similar to the scope of this bid; documents must be provided within five business days of request.
16. Request additional documentation from the Bidder and to request reports on financial stability from independent financial rating services.
17. Reject any Bidder who does not demonstrate financial stability sufficient for the scope of this bid.

**APPENDIX A**

**STANDARD CLAUSES FOR NEW YORK STATE CONTRACTS**

**PLEASE RETAIN THIS DOCUMENT  
FOR FUTURE REFERENCE**

## **TABLE OF CONTENTS**

		<b>Page</b>
1.	<b>Executory Clause</b>	3
2.	<b>Non-Assignment Clause</b>	3
3.	<b>Comptroller’s Approval</b>	3
4.	<b>Workers’ Compensation Benefits</b>	3
5.	<b>Non-Discrimination Requirements</b>	3
6.	<b>Wage and Hours Provisions</b>	3
7.	<b>Non-Collusive Bidding Certification</b>	4
8.	<b>International Boycott Prohibition</b>	4
9.	<b>Set-Off Rights</b>	4
10.	<b>Records</b>	4
11.	<b>Identifying Information and Privacy Notification</b>	4
12.	<b>Equal Employment Opportunities For Minorities and Women</b>	4
13.	<b>Conflicting Terms</b>	5
14.	<b>Governing Law</b>	5
15.	<b>Late Payment</b>	5
16.	<b>No Arbitration</b>	5
17.	<b>Service of Process</b>	5
18.	<b>Prohibition on Purchase of Tropical Hardwoods</b>	5
19.	<b>MacBride Fair Employment Principles</b>	6
20.	<b>Omnibus Procurement Act of 1992</b>	6
21.	<b>Reciprocity and Sanctions Provisions</b>	6
22.	<b>Compliance with New York State Information Security Breach and Notification Act</b>	6
23.	<b>Compliance with Consultant Disclosure Law</b>	6
24.	<b>Procurement Lobbying</b>	6
25.	<b>Certification of Registration to Collect Sales and Compensating Use Tax by Certain State Contractors, Affiliates and Subcontractors</b>	7

## **STANDARD CLAUSES FOR NYS CONTRACTS**

The parties to the attached contract, license, lease, amendment or other agreement of any kind (hereinafter, "the contract" or "this contract") agree to be bound by the following clauses which are hereby made a part of the contract (the word "Contractor" herein refers to any party other than the State, whether a contractor, licensor, licensee, lessor, lessee or any other party):

**1. EXECUTORY CLAUSE.** In accordance with Section 41 of the State Finance Law, the State shall have no liability under this contract to the Contractor or to anyone else beyond funds appropriated and available for this contract.

**2. NON-ASSIGNMENT CLAUSE.** In accordance with Section 138 of the State Finance Law, this contract may not be assigned by the Contractor or its right, title or interest therein assigned, transferred, conveyed, sublet or otherwise disposed of without the State's previous written consent, and attempts to do so are null and void. Notwithstanding the foregoing, such prior written consent of an assignment of a contract let pursuant to Article XI of the State Finance Law may be waived at the discretion of the contracting agency and with the concurrence of the State Comptroller where the original contract was subject to the State Comptroller's approval, where the assignment is due to a reorganization, merger or consolidation of the Contractor's business entity or enterprise. The State retains its right to approve an assignment and to require that any Contractor demonstrate its responsibility to do business with the State. The Contractor may, however, assign its right to receive payments without the State's prior written consent unless this contract concerns Certificates of Participation pursuant to Article 5-A of the State Finance Law.

**3. COMPTROLLER'S APPROVAL.** In accordance with Section 112 of the State Finance Law (or, if this contract is with the State University or City University of New York, Section 355 or Section 6218 of the Education Law), if this contract exceeds \$50,000 (or the minimum thresholds agreed to by the Office of the State Comptroller for certain S.U.N.Y. and C.U.N.Y. contracts), or if this is an amendment for any amount to a contract which, as so amended, exceeds said statutory amount, or if, by this contract, the State agrees to give something other than money when the value or reasonably estimated value of such consideration exceeds \$10,000, it shall not be valid, effective or binding upon the State until it has been approved by the State Comptroller and filed in his office. Comptroller's approval of contracts let by the Office of General Services is required when such contracts exceed \$85,000 (State Finance Law Section 163.6.a).

**4. WORKERS' COMPENSATION BENEFITS.** In accordance with Section 142 of the State Finance Law, this contract shall be void and of no force and effect unless the

Contractor shall provide and maintain coverage during the life of this contract for the benefit of such employees as are required to be covered by the provisions of the Workers' Compensation Law.

**5. NON-DISCRIMINATION REQUIREMENTS.** To the extent required by Article 15 of the Executive Law (also known as the Human Rights Law) and all other State and Federal statutory and constitutional non-discrimination provisions, the Contractor will not discriminate against any employee or applicant for employment because of race, creed, color, sex, national origin, sexual orientation, age, disability, genetic predisposition or carrier status, or marital status. Furthermore, in accordance with Section 220-e of the Labor Law, if this is a contract for the construction, alteration or repair of any public building or public work or for the manufacture, sale or distribution of materials, equipment or supplies, and to the extent that this contract shall be performed within the State of New York, Contractor agrees that neither it nor its subcontractors shall, by reason of race, creed, color, disability, sex, or national origin: (a) discriminate in hiring against any New York State citizen who is qualified and available to perform the work; or (b) discriminate against or intimidate any employee hired for the performance of work under this contract. If this is a building service contract as defined in Section 230 of the Labor Law, then, in accordance with Section 239 thereof, Contractor agrees that neither it nor its subcontractors shall by reason of race, creed, color, national origin, age, sex or disability: (a) discriminate in hiring against any New York State citizen who is qualified and available to perform the work; or (b) discriminate against or intimidate any employee hired for the performance of work under this contract. Contractor is subject to fines of \$50.00 per person per day for any violation of Section 220-e or Section 239 as well as possible termination of this contract and forfeiture of all moneys due hereunder for a second or subsequent violation.

**6. WAGE AND HOURS PROVISIONS.** If this is a public work contract covered by Article 8 of the Labor Law or a building service contract covered by Article 9 thereof, neither Contractor's employees nor the employees of its subcontractors may be required or permitted to work more than the number of hours or days stated in said statutes, except as otherwise provided in the Labor Law and as set forth in prevailing wage and supplement schedules issued by the State Labor Department. Furthermore, Contractor and its subcontractors must pay at least the prevailing wage rate and pay or provide the prevailing supplements, including the premium rates for overtime pay, as determined by the State Labor Department in accordance with the Labor Law. Additionally, effective April 28, 2008, if this is a public work contract covered by Article 8 of the Labor Law, the Contractor understands and agrees that the filing of payrolls in a manner consistent with Subdivision 3-a of Section 220 of the Labor

Law shall be a condition precedent to payment by the State of any State approved sums due and owing for work done upon the project.

**7. NON-COLLUSIVE BIDDING CERTIFICATION.** In accordance with Section 139-d of the State Finance Law, if this contract was awarded based upon the submission of bids, Contractor affirms, under penalty of perjury, that its bid was arrived at independently and without collusion aimed at restricting competition. Contractor further affirms that, at the time Contractor submitted its bid, an authorized and responsible person executed and delivered to the State a non-collusive bidding certification on Contractor's behalf.

**8. INTERNATIONAL BOYCOTT PROHIBITION.** In accordance with Section 220-f of the Labor Law and Section 139-h of the State Finance Law, if this contract exceeds \$5,000, the Contractor agrees, as a material condition of the contract, that neither the Contractor nor any substantially owned or affiliated person, firm, partnership or corporation has participated, is participating, or shall participate in an international boycott in violation of the federal Export Administration Act of 1979 (50 USC App. Sections 2401 et seq.) or regulations thereunder. If such Contractor, or any of the aforesaid affiliates of Contractor, is convicted or is otherwise found to have violated said laws or regulations upon the final determination of the United States Commerce Department or any other appropriate agency of the United States subsequent to the contract's execution, such contract, amendment or modification thereto shall be rendered forfeit and void. The Contractor shall so notify the State Comptroller within five (5) business days of such conviction, determination or disposition of appeal (2NYCRR 105.4).

**9. SET-OFF RIGHTS.** The State shall have all of its common law, equitable and statutory rights of set-off. These rights shall include, but not be limited to, the State's option to withhold for the purposes of set-off any moneys due to the Contractor under this contract up to any amounts due and owing to the State with regard to this contract, any other contract with any State department or agency, including any contract for a term commencing prior to the term of this contract, plus any amounts due and owing to the State for any other reason including, without limitation, tax delinquencies, fee delinquencies or monetary penalties relative thereto. The State shall exercise its set-off rights in accordance with normal State practices including, in cases of set-off pursuant to an audit, the finalization of such audit by the State agency, its representatives, or the State Comptroller.

**10. RECORDS.** The Contractor shall establish and maintain complete and accurate books, records, documents, accounts and other evidence directly pertinent to performance under this contract (hereinafter, collectively, "the Records"). The Records must be kept for the balance of the calendar year in which they were made and for six (6) additional years thereafter. The State Comptroller, the Attorney General and any other person or entity authorized to conduct an examination, as well as the agency or agencies involved in this contract, shall have access to the Records during normal business hours at an office of the Contractor within the State of New York or, if no such office is

available, at a mutually agreeable and reasonable venue within the State, for the term specified above for the purposes of inspection, auditing and copying. The State shall take reasonable steps to protect from public disclosure any of the Records which are exempt from disclosure under Section 87 of the Public Officers Law (the "Statute") provided that: (i) the Contractor shall timely inform an appropriate State official, in writing, that said records should not be disclosed; and (ii) said records shall be sufficiently identified; and (iii) designation of said records as exempt under the Statute is reasonable. Nothing contained herein shall diminish, or in any way adversely affect, the State's right to discovery in any pending or future litigation.

**11. IDENTIFYING INFORMATION AND PRIVACY NOTIFICATION.** (a) Identification Number(s). Every invoice or New York State Claim for Payment submitted to a New York State agency by a payee, for payment for the sale of goods or services or for transactions (e.g., leases, easements, licenses, etc.) related to real or personal property must include the payee's identification number. The number is any or all of the following: (i) the payee's Federal employer identification number, (ii) the payee's Federal social security number, and/or (iii) the payee's Vendor Identification Number assigned by the Statewide Financial System. Failure to include such number or numbers may delay payment. Where the payee does not have such number or numbers, the payee, on its invoice or Claim for Payment, must give the reason or reasons why the payee does not have such number or numbers.

(b) Privacy Notification. (1) The authority to request the above personal information from a seller of goods or services or a lessor of real or personal property, and the authority to maintain such information, is found in Section 5 of the State Tax Law. Disclosure of this information by the seller or lessor to the State is mandatory. The principal purpose for which the information is collected is to enable the State to identify individuals, businesses and others who have been delinquent in filing tax returns or may have understated their tax liabilities and to generally identify persons affected by the taxes administered by the Commissioner of Taxation and Finance. The information will be used for tax administration purposes and for any other purpose authorized by law. (2) The personal information is requested by the purchasing unit of the agency contracting to purchase the goods or services or lease the real or personal property covered by this contract or lease. The information is maintained in the Statewide Financial System by the Vendor Management Unit within the Bureau of State Expenditures, Office of the State Comptroller, 110 State Street, Albany, New York 12236.

**12. EQUAL EMPLOYMENT OPPORTUNITIES FOR MINORITIES AND WOMEN.** In accordance with Section 312 of the Executive Law and 5 NYCRR 143, if this contract is: (i) a written agreement or purchase order instrument, providing for a total expenditure in excess of \$25,000.00, whereby a contracting agency is committed to expend or does expend funds in return for labor, services, supplies, equipment, materials or any combination of the foregoing, to be performed for, or rendered or furnished to the contracting agency; or (ii) a written agreement in excess of \$100,000.00 whereby a contracting agency is committed to expend or does

expend funds for the acquisition, construction, demolition, replacement, major repair or renovation of real property and improvements thereon; or (iii) a written agreement in excess of \$100,000.00 whereby the owner of a State assisted housing project is committed to expend or does expend funds for the acquisition, construction, demolition, replacement, major repair or renovation of real property and improvements thereon for such project, then the following shall apply and by signing this agreement the Contractor certifies and affirms that it is Contractor's equal employment opportunity policy that:

(a) The Contractor will not discriminate against employees or applicants for employment because of race, creed, color, national origin, sex, age, disability or marital status, shall make and document its conscientious and active efforts to employ and utilize minority group members and women in its work force on State contracts and will undertake or continue existing programs of affirmative action to ensure that minority group members and women are afforded equal employment opportunities without discrimination. Affirmative action shall mean recruitment, employment, job assignment, promotion, upgradings, demotion, transfer, layoff, or termination and rates of pay or other forms of compensation;

(b) at the request of the contracting agency, the Contractor shall request each employment agency, labor union, or authorized representative of workers with which it has a collective bargaining or other agreement or understanding, to furnish a written statement that such employment agency, labor union or representative will not discriminate on the basis of race, creed, color, national origin, sex, age, disability or marital status and that such union or representative will affirmatively cooperate in the implementation of the Contractor's obligations herein; and

(c) the Contractor shall state, in all solicitations or advertisements for employees, that, in the performance of the State contract, all qualified applicants will be afforded equal employment opportunities without discrimination because of race, creed, color, national origin, sex, age, disability or marital status.

Contractor will include the provisions of "a", "b", and "c" above, in every subcontract over \$25,000.00 for the construction, demolition, replacement, major repair, renovation, planning or design of real property and improvements thereon (the "Work") except where the Work is for the beneficial use of the Contractor. Section 312 does not apply to: (i) work, goods or services unrelated to this contract; or (ii) employment outside New York State. The State shall consider compliance by a contractor or subcontractor with the requirements of any federal law concerning equal employment opportunity which effectuates the purpose of this section. The contracting agency shall determine whether the imposition of the requirements of the provisions hereof duplicate or conflict with any such federal law and if such duplication or conflict exists, the contracting agency shall waive the applicability of Section 312 to the extent of such duplication or conflict. Contractor will comply with all duly promulgated and lawful rules and regulations of the Department of Economic

Development's Division of Minority and Women's Business Development pertaining hereto.

**13. CONFLICTING TERMS.** In the event of a conflict between the terms of the contract (including any and all attachments thereto and amendments thereof) and the terms of this Appendix A, the terms of this Appendix A shall control.

**14. GOVERNING LAW.** This contract shall be governed by the laws of the State of New York except where the Federal supremacy clause requires otherwise.

**15. LATE PAYMENT.** Timeliness of payment and any interest to be paid to Contractor for late payment shall be governed by Article 11-A of the State Finance Law to the extent required by law.

**16. NO ARBITRATION.** Disputes involving this contract, including the breach or alleged breach thereof, may not be submitted to binding arbitration (except where statutorily authorized), but must, instead, be heard in a court of competent jurisdiction of the State of New York.

**17. SERVICE OF PROCESS.** In addition to the methods of service allowed by the State Civil Practice Law & Rules ("CPLR"), Contractor hereby consents to service of process upon it by registered or certified mail, return receipt requested. Service hereunder shall be complete upon Contractor's actual receipt of process or upon the State's receipt of the return thereof by the United States Postal Service as refused or undeliverable. Contractor must promptly notify the State, in writing, of each and every change of address to which service of process can be made. Service by the State to the last known address shall be sufficient. Contractor will have thirty (30) calendar days after service hereunder is complete in which to respond.

**18. PROHIBITION ON PURCHASE OF TROPICAL HARDWOODS.** The Contractor certifies and warrants that all wood products to be used under this contract award will be in accordance with, but not limited to, the specifications and provisions of Section 165 of the State Finance Law, (Use of Tropical Hardwoods) which prohibits purchase and use of tropical hardwoods, unless specifically exempted, by the State or any governmental agency or political subdivision or public benefit corporation. Qualification for an exemption under this law will be the responsibility of the contractor to establish to meet with the approval of the State.

In addition, when any portion of this contract involving the use of woods, whether supply or installation, is to be performed by any subcontractor, the prime Contractor will indicate and certify in the submitted bid proposal that the subcontractor has been informed and is in compliance with specifications and provisions regarding use of tropical hardwoods as detailed in §165 State Finance Law. Any such use must meet with the approval of the State; otherwise, the bid may not be considered responsive. Under bidder certifications, proof of qualification for exemption will be the responsibility of the Contractor to meet with the approval of the State.

**19. MACBRIDE FAIR EMPLOYMENT PRINCIPLES.**

In accordance with the MacBride Fair Employment Principles (Chapter 807 of the Laws of 1992), the Contractor hereby stipulates that the Contractor either (a) has no business operations in Northern Ireland, or (b) shall take lawful steps in good faith to conduct any business operations in Northern Ireland in accordance with the MacBride Fair Employment Principles (as described in Section 165 of the New York State Finance Law), and shall permit independent monitoring of compliance with such principles.

**20. OMNIBUS PROCUREMENT ACT OF 1992.** It is the policy of New York State to maximize opportunities for the participation of New York State business enterprises, including minority and women-owned business enterprises as bidders, subcontractors and suppliers on its procurement contracts.

Information on the availability of New York State subcontractors and suppliers is available from:

NYS Department of Economic Development  
 Division for Small Business  
 30 South Pearl St -- 7<sup>th</sup> Floor  
 Albany, New York 12245  
 Telephone: 518-292-5220  
 Fax: 518-292-5884  
<http://www.empire.state.ny.us>

A directory of certified minority and women-owned business enterprises is available from:

NYS Department of Economic Development  
 Division of Minority and Women's Business Development  
 30 South Pearl St -- 2nd Floor  
 Albany, New York 12245  
 Telephone: 518-292-5250  
 Fax: 518-292-5803  
<http://www.empire.state.ny.us>

The Omnibus Procurement Act of 1992 requires that by signing this bid proposal or contract, as applicable, Contractors certify that whenever the total bid amount is greater than \$1 million:

- (a) The Contractor has made reasonable efforts to encourage the participation of New York State Business Enterprises as suppliers and subcontractors, including certified minority and women-owned business enterprises, on this project, and has retained the documentation of these efforts to be provided upon request to the State;
- (b) The Contractor has complied with the Federal Equal Opportunity Act of 1972 (P.L. 92-261), as amended;
- (c) The Contractor agrees to make reasonable efforts to provide notification to New York State residents of employment opportunities on this project through listing any such positions with the Job Service Division of the New York

State Department of Labor, or providing such notification in such manner as is consistent with existing collective bargaining contracts or agreements. The Contractor agrees to document these efforts and to provide said documentation to the State upon request; and

- (d) The Contractor acknowledges notice that the State may seek to obtain offset credits from foreign countries as a result of this contract and agrees to cooperate with the State in these efforts.

**21. RECIPROCITY AND SANCTIONS PROVISIONS.**

Bidders are hereby notified that if their principal place of business is located in a country, nation, province, state or political subdivision that penalizes New York State vendors, and if the goods or services they offer will be substantially produced or performed outside New York State, the Omnibus Procurement Act 1994 and 2000 amendments (Chapter 684 and Chapter 383, respectively) require that they be denied contracts which they would otherwise obtain. NOTE: As of May 15, 2002, the list of discriminatory jurisdictions subject to this provision includes the states of South Carolina, Alaska, West Virginia, Wyoming, Louisiana and Hawaii. Contact NYS Department of Economic Development for a current list of jurisdictions subject to this provision.

**22. COMPLIANCE WITH NEW YORK STATE INFORMATION SECURITY BREACH AND NOTIFICATION ACT.**

Contractor shall comply with the provisions of the New York State Information Security Breach and Notification Act (General Business Law Section 899-aa; State Technology Law Section 208).

**23. COMPLIANCE WITH CONSULTANT DISCLOSURE LAW.**

If this is a contract for consulting services, defined for purposes of this requirement to include analysis, evaluation, research, training, data processing, computer programming, engineering, environmental, health, and mental health services, accounting, auditing, paralegal, legal or similar services, then, in accordance with Section 163 (4-g) of the State Finance Law (as amended by Chapter 10 of the Laws of 2006), the Contractor shall timely, accurately and properly comply with the requirement to submit an annual employment report for the contract to the agency that awarded the contract, the Department of Civil Service and the State Comptroller.

**24. PROCUREMENT LOBBYING.**

To the extent this agreement is a "procurement contract" as defined by State Finance Law Sections 139-j and 139-k, by signing this agreement the contractor certifies and affirms that all disclosures made in accordance with State Finance Law Sections 139-j and 139-k are complete, true and accurate. In the event such certification is found to be intentionally false or intentionally incomplete, the State may terminate the agreement by providing written notification to the Contractor in accordance with the terms of the agreement.

**25. CERTIFICATION OF REGISTRATION TO COLLECT SALES AND COMPENSATING USE TAX BY CERTAIN STATE CONTRACTORS, AFFILIATES AND SUBCONTRACTORS.**

To the extent this agreement is a contract as defined by Tax Law Section 5-a, if the contractor fails to make the certification required by Tax Law Section 5-a or if during the

term of the contract, the Department of Taxation and Finance or the covered agency, as defined by Tax Law 5-a, discovers that the certification, made under penalty of perjury, is false, then such failure to file or false certification shall be a material breach of this contract and this contract may be terminated, by providing written notification to the Contractor in accordance with the terms of the agreement, if the covered agency determines that such action is in the best interest of the State.

# **APPENDIX B**

## **REQUIRED FORMS**

- **QUOTE PROPOSAL FORM**
- **CONTRACTOR INFORMATION**
- **MACBRIDE FAIR EMPLOYMENT PRINCIPLES**
- **NON-COLLUSIVE BIDDING CERTIFICATION**
- **ST-220-TD TAX & FINANCE CONTRACTOR CERTIFICATION**
- **ST-220-CA TAX & FINANCE AGENCY CERTIFICATION**
- **CONSULTANT CONTRACTOR FORMS**
- **SUBSTITUTE W-9**

**SOLICITATION #1768**  
**QUOTE PROPOSAL FORM – IT SERVICES TO DEVELOP A WEB-BASED APPLICATION “eHBITS”**

Company’s Name: \_\_\_\_\_

Address: \_\_\_\_\_

\_\_\_\_\_

Phone #: \_\_\_\_\_

Email Address: \_\_\_\_\_

The above contractor agrees to perform IT Services to Develop a Web-Based Application “eHBITS”, as outlined in the specifications of this RFQ, for the NYS Office of General Services, Information Resource Management.

The Contractor may not propose more than four (4) individuals when responding to this RFQ.

Consultant Name	Job Title	Estimated Hours	Hourly Bill Rate	Total (Estimated Hours x Hourly Bill Rate/Hour)
		<b>NOT TO EXCEED GRAND TOTAL</b>		

**Contractor Information**

**SOLICITATION NUMBER 1768**

Offerer affirms that it understands and agrees to comply with the procedures of the Government Entity relative to permissible contacts as required by New York State Finance Law §139-j (3) and §139-j (6) (b).

\_\_\_\_\_  
(Authorized Signature)

\_\_\_\_\_  
(Date)

\_\_\_\_\_  
(Print Name)

\_\_\_\_\_  
(Title)

\_\_\_\_\_  
(Company Name)

\_\_\_\_\_  
(Federal I.D. Number)

\_\_\_\_\_  
(NYS Vendor I.D. Number)

\_\_\_\_\_  
(Address)

\_\_\_\_\_  
(City, State, Zip)

\_\_\_\_\_  
(County)

\_\_\_\_\_  
(Telephone Number) Ext. \_\_\_\_\_

\_\_\_\_\_  
(Toll Free Phone) Ext. \_\_\_\_\_

\_\_\_\_\_  
(Fax Number)

\_\_\_\_\_  
(Toll Free Fax Number)

\_\_\_\_\_  
(E-mail)

New York State Small Business  Yes  No

New York State Certified Minority Owned Business  Yes  No

New York State Certified Woman Owned Business  Yes  No

Do you understand and is your firm capable of meeting the insurance requirements to enter into a contract with New York State?

Yes  No

Does your proposal meet all the requirements of this solicitation?

Yes  No

---

**BIDDER/OFFERER DISCLOSURE OF PRIOR NON-RESPONSIBILITY DETERMINATIONS**

Pursuant to Procurement Lobbying Law (SFL §139-j)

A. Has any Governmental Entity made a finding of non-responsibility regarding the individual or entity seeking to enter into the Procurement Contract in the previous four years?

\_\_\_\_\_ YES \_\_\_\_\_ NO

If yes, please answer the following question:

B. Was the basis for the finding of non-responsibility due to a violation of State Finance Law §139-j?

\_\_\_\_\_ YES \_\_\_\_\_ NO

C. If yes, was the basis for the finding of non-responsibility due to the intentional provision of false or incomplete information to a governmental entity?

\_\_\_\_\_ YES \_\_\_\_\_ NO

If yes, please provide details regarding the finding of non-responsibility:

Governmental Entity:

\_\_\_\_\_

Date of Finding of Non-responsibility:

\_\_\_\_\_

Basis of Finding of Non-Responsibility:  
(add additional pages if necessary)

\_\_\_\_\_

\_\_\_\_\_

D. Has any governmental agency terminated or withheld a procurement contract with the above-named individual or entity due to the intentional provision of false or incomplete information?

\_\_\_\_\_ YES \_\_\_\_\_ NO

If yes, please provide details:

Governmental Entity:

\_\_\_\_\_

Date of Termination or Withholding of Contract:

\_\_\_\_\_

Basis of Termination or Withholding:  
(add additional pages if necessary)

\_\_\_\_\_

\_\_\_\_\_

**Bidder is required to sign both sections on this page**

**MacBride Fair Employment Principles**

**NONDISCRIMINATION IN EMPLOYMENT IN NORTHERN IRELAND:  
MACBRIDE FAIR EMPLOYMENT PRINCIPLES**

In accordance with Chapter 807 of the Laws of 1992 the bidder, by submission of this bid, certifies that it or any individual or legal entity in which the bidder holds a 10% or greater ownership interest, or any individual or legal entity that holds a 10% or greater ownership interest in the bidder, either: (answer yes or no to one or both of the following, as applicable:

1. Have business operations in Northern Ireland,

Yes  No

If yes:

2. Shall take lawful steps in good faith to conduct any business operations they have in Northern Ireland in accordance with the MacBride Fair Employment Principles relating to nondiscrimination in employment and freedom of workplace opportunity regarding such operations in Northern Ireland, and shall permit independent monitoring of their compliance with such Principles.

Yes  No

\_\_\_\_\_  
(Contractor's Signature)

\_\_\_\_\_  
(Name of Business)

**Non-Collusive Bidding Certification**

By submission of this bid, each bidder and each person signing on behalf of any bidder certifies, and in the case of a joint bid each party thereto certifies as to its own organization, under penalty of perjury, that to the best of his knowledge and belief) The prices in this bid have been arrived at independently without collusion, consultation, communication, or agreement, for the purpose of restricting competition, as to any matter relating to such prices with any other bidder or with any competitor) Unless otherwise required by law, the prices which have been quoted in this bid have not been knowingly disclosed by the bidder and will not knowingly be disclosed by the bidder prior to opening, directly or indirectly, to any other bidder or to any competitor; an) No attempt has been made or will be made by the bidder to induce any other person, partnership or corporation to submit or not to submit a bid for the purpose of restricting competition.

\_\_\_\_\_  
(Contractor's Signature)

\_\_\_\_\_  
(Name of Business)

# **New York State Department of Taxation and Finance**

## **Contractor Certification (ST-220-TD)**

*(If filing with the Department of Taxation & Finance for the first time, or previously submitted information needs to be updated, these 4 pages must be removed from this RFQ, completed, signed and submitted directly to the Department of Taxation and Finance.)*

## **Contractor Certification to Covered Agency (ST-220-CA)**

*(Regardless of whether ST-220-TD is being filed/updated for this RFQ or not, these 2 pages must be completed, signed and returned with this RFQ.)*



# Contractor Certification

(Pursuant to Section 5-a of the Tax Law, as amended, effective April 26, 2006)

# ST-220-TD

(6/06)

For information, consult Publication 223, *Questions and Answers Concerning Tax Law Section 5-a* (see *Need help?* below).

Contractor name		
Contractor's principal place of business	City	State ZIP code
Contractor's mailing address (if different than above)		
Contractor's federal employer identification number (EIN)	Contractor's sales tax ID number (if different from contractor's EIN)	Contractor's telephone number ( )
Covered agency name	Contract number or description	Estimated contract value over the full term of contract (but not including renewals) \$
Covered agency address	Covered agency telephone number	

### General information

Section 5-a of the Tax Law, as amended, effective April 26, 2006, requires certain contractors awarded certain state contracts valued at more than \$100,000 to certify to the Tax Department that they are registered to collect New York State and local sales and compensating use taxes, if they made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000, measured over a specified period. In addition, contractors must certify to the Tax Department that each affiliate and subcontractor exceeding such sales threshold during a specified period is registered to collect New York State and local sales and compensating use taxes. Contractors must also file a Form ST-220-CA, certifying to the procuring state entity that they filed Form ST-220-TD with the Tax Department and that the information contained on Form ST-220-TD is correct and complete as of the date they file Form ST-220-CA.

For more detailed information regarding this form and section 5-a of the Tax Law, see Publication 223, *Questions and Answers Concerning Tax Law Section 5-a*, (as amended, effective April 26, 2006), available at [www.nystax.gov](http://www.nystax.gov). Information is also available by calling the Tax Department's Contractor Information Center at 1 800 698-2931.

**Note:** Form ST-220-TD must be signed by a person authorized to make the certification on behalf of the contractor, and the acknowledgement on page 4 of this form must be completed before a notary public.

Mail completed form to:

**NYS TAX DEPARTMENT  
DATA ENTRY SECTION  
W A HARRIMAN CAMPUS  
ALBANY NY 12227**

### Privacy notification

The Commissioner of Taxation and Finance may collect and maintain personal information pursuant to the New York State Tax Law, including but not limited to, sections 5-a, 171, 171-a, 287, 308, 429, 475, 505, 697, 1096, 1142, and 1415 of that Law; and may require disclosure of social security numbers pursuant to 42 USC 405(c)(2)(C)(i).

This information will be used to determine and administer tax liabilities and, when authorized by law, for certain tax offset and exchange of tax information programs as well as for any other lawful purpose.

Information concerning quarterly wages paid to employees is provided to certain state agencies for purposes of fraud prevention, support enforcement, evaluation of the effectiveness of certain employment and training programs and other purposes authorized by law.

Failure to provide the required information may subject you to civil or criminal penalties, or both, under the Tax Law.

This information is maintained by the Director of Records Management and Data Entry, NYS Tax Department, W A Harriman Campus, Albany NY 12227; telephone 1 800 225-5829. From areas outside the United States and outside Canada, call (518) 485-6800.

### Need help?

 **Internet access:** [www.nystax.gov](http://www.nystax.gov)  
(for information, forms, and publications)

 **Fax-on-demand forms:** 1 800 748-3676

 **Telephone assistance** is available from 8:00 A.M. to 5:00 P.M. (eastern time), Monday through Friday.

To order forms and publications: 1 800 462-8100

Sales Tax information Center: 1 800 698-2909

From areas outside the U.S. and outside Canada: (518) 485-6800

**Hearing and speech impaired** (telecommunications device for the deaf (TDD) callers only): 1 800 634-2110

 **Persons with disabilities:** In compliance with the Americans with Disabilities Act, we will ensure that our lobbies, offices, meeting rooms, and other facilities are accessible to persons with disabilities. If you have questions about special accommodations for persons with disabilities, please call 1 800 972-1233.

I, \_\_\_\_\_, hereby affirm, under penalty of perjury, that I am \_\_\_\_\_  
(name) (title)

of the above-named contractor, and that I am authorized to make this certification on behalf of such contractor.

Make only one entry in each section below.

**Section 1 — Contractor registration status**

- The contractor has made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000 during the four sales tax quarters which immediately precede the sales tax quarter in which this certification is made. The contractor is registered to collect New York State and local sales and compensating use taxes with the Commissioner of Taxation and Finance pursuant to sections 1134 and 1253 of the Tax Law, and is listed on Schedule A of this certification.
- The contractor has not made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000 during the four sales tax quarters which immediately precede the sales tax quarter in which this certification is made.

**Section 2 — Affiliate registration status**

- The contractor does not have any affiliates.
- To the best of the contractor's knowledge, the contractor has one or more affiliates having made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000 during the four sales tax quarters which immediately precede the sales tax quarter in which this certification is made, and each affiliate exceeding the \$300,000 cumulative sales threshold during such quarters is registered to collect New York State and local sales and compensating use taxes with the Commissioner of Taxation and Finance pursuant to sections 1134 and 1253 of the Tax Law. The contractor has listed each affiliate exceeding the \$300,000 cumulative sales threshold during such quarters on Schedule A of this certification.
- To the best of the contractor's knowledge, the contractor has one or more affiliates, and each affiliate has not made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000 during the four sales tax quarters which immediately precede the sales tax quarter in which this certification is made.

**Section 3 — Subcontractor registration status**

- The contractor does not have any subcontractors.
- To the best of the contractor's knowledge, the contractor has one or more subcontractors having made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000 during the four sales tax quarters which immediately precede the sales tax quarter in which this certification is made, and each subcontractor exceeding the \$300,000 cumulative sales threshold during such quarters is registered to collect New York State and local sales and compensating use taxes with the Commissioner of Taxation and Finance pursuant to sections 1134 and 1253 of the Tax Law. The contractor has listed each subcontractor exceeding the \$300,000 cumulative sales threshold during such quarters on Schedule A of this certification.
- To the best of the contractor's knowledge, the contractor has one or more subcontractors, and each subcontractor has not made sales delivered by any means to locations within New York State of tangible personal property or taxable services having a cumulative value in excess of \$300,000 during the four sales tax quarters which immediately precede the sales tax quarter in which this certification is made.

Sworn to this \_\_\_ day of \_\_\_\_\_, 20 \_\_\_\_

\_\_\_\_\_  
(sign before a notary public)

\_\_\_\_\_  
(title)







# Contractor Certification to Covered Agency

(Pursuant to Section 5-a of the Tax Law, as amended, effective April 26, 2006)

# ST-220-CA

(6/06)

For information, consult Publication 223, *Questions and Answers Concerning Tax Law Section 5-a* (see *Need Help?* on back).

Contractor name				For covered agency use only Contract number or description	
Contractor's principal place of business		City	State	ZIP code	
Contractor's mailing address (if different than above)					
Contractor's federal employer identification number (EIN)			Contractor's sales tax ID number (if different from contractor's EIN)		
Estimated contract value over the full term of contract (but not including renewals)					
\$					
Contractor's telephone number		Covered agency name			
Covered agency address					Covered agency telephone number

I, \_\_\_\_\_, hereby affirm, under penalty of perjury, that I am \_\_\_\_\_  
*(name)* *(title)*

of the above-named contractor, that I am authorized to make this certification on behalf of such contractor, and I further certify that:

*(Mark an X in only one box)*

The contractor has filed Form ST-220-TD with the Department of Taxation and Finance in connection with this contract and, to the best of contractor's knowledge, the information provided on the Form ST-220-TD, is correct and complete.

The contractor has previously filed Form ST-220-TD with the Tax Department in connection with \_\_\_\_\_  
*(insert contract number or description)*

and, to the best of the contractor's knowledge, the information provided on that previously filed Form ST-220-TD, is correct and complete as of the current date, and thus the contractor is not required to file a new Form ST-220-TD at this time.

Sworn to this \_\_\_\_ day of \_\_\_\_\_, 20 \_\_\_\_

\_\_\_\_\_  
*(sign before a notary public)*

\_\_\_\_\_  
*(title)*

## Instructions

### General information

Tax Law section 5-a was amended, effective April 26, 2006. On or after that date, in all cases where a contract is subject to Tax Law section 5-a, a contractor must file (1) Form ST-220-CA, *Contractor Certification to Covered Agency*, with a covered agency, and (2) Form ST-220-TD with the Tax Department before a contract may take effect. The circumstances when a contract is subject to section 5-a are listed in Publication 223, Q&A 3. This publication is available on our Web site, by fax, or by mail. (See *Need help?* for more information on how to obtain this publication.) In addition, a contractor must file a new Form ST-220-CA with a covered agency before an existing contract with such agency may be renewed.

If you have questions, please call our information center at 1 800 698-2931.

**Note:** Form ST-220-CA must be signed by a person authorized to make the certification on behalf of the contractor, and the acknowledgement on page 2 of this form must be completed before a notary public.

### When to complete this form

As set forth in Publication 223, a contract is subject to section 5-a, and you must make the required certification(s), if:

- i. The procuring entity is a *covered agency* within the meaning of the statute (see Publication 223, Q&A 5);
- ii. The contractor is a *contractor* within the meaning of the statute (see Publication 223, Q&A 6); and
- iii. The contract is a *contract* within the meaning of the statute. This is the case when it (a) has a value in excess of \$100,000 and (b) is a contract for *commodities* or *services*, as such terms are defined for purposes of the statute (see Publication 223, Q&A 8 and 9).

Furthermore, the procuring entity must have begun the solicitation to purchase on or after January 1, 2005, and the resulting contract must have been awarded, amended, extended, renewed, or assigned on or after April 26, 2006 (the effective date of the section 5-a amendments).

Individual, Corporation, Partnership, or LLC Acknowledgment

STATE OF }
: SS.:
COUNTY OF }

On the \_\_\_ day of \_\_\_\_\_ in the year 20\_\_\_, before me personally appeared \_\_\_\_\_,
known to me to be the person who executed the foregoing instrument, who, being duly sworn by me did depose and say that
\_he resides at \_\_\_\_\_,
Town of \_\_\_\_\_,
County of \_\_\_\_\_,
State of \_\_\_\_\_; and further that:

[Mark an X in the appropriate box and complete the accompanying statement.]

- (If an individual): \_he executed the foregoing instrument in his/her name and on his/her own behalf.
(If a corporation): \_he is the \_\_\_\_\_ of \_\_\_\_\_, the corporation described in said instrument; that, by authority of the Board of Directors of said corporation, \_he is authorized to execute the foregoing instrument on behalf of the corporation for purposes set forth therein; and that, pursuant to that authority, \_he executed the foregoing instrument in the name of and on behalf of said corporation as the act and deed of said corporation.
(If a partnership): \_he is a \_\_\_\_\_ of \_\_\_\_\_, the partnership described in said instrument; that, by the terms of said partnership, \_he is authorized to execute the foregoing instrument on behalf of the partnership for purposes set forth therein; and that, pursuant to that authority, \_he executed the foregoing instrument in the name of and on behalf of said partnership as the act and deed of said partnership.
(If a limited liability company): \_he is a duly authorized member of \_\_\_\_\_, LLC, the limited liability company described in said instrument; that \_he is authorized to execute the foregoing instrument on behalf of the limited liability company for purposes set forth therein; and that, pursuant to that authority, \_he executed the foregoing instrument in the name of and on behalf of said limited liability company as the act and deed of said limited liability company.

Notary Public

Registration No.

Privacy notification

The Commissioner of Taxation and Finance may collect and maintain personal information pursuant to the New York State Tax Law, including but not limited to, sections 5-a, 171, 171-a, 287, 308, 429, 475, 505, 687, 1096, 1142, and 1415 of that Law; and may require disclosure of social security numbers pursuant to 42 USC 405(c)(2)(C)(i).

This information will be used to determine and administer tax liabilities and, when authorized by law, for certain tax offset and exchange of tax information programs as well as for any other lawful purpose.

Information concerning quarterly wages paid to employees is provided to certain state agencies for purposes of fraud prevention, support enforcement, evaluation of the effectiveness of certain employment and training programs and other purposes authorized by law.

Failure to provide the required information may subject you to civil or criminal penalties, or both, under the Tax Law.

This information is maintained by the Director of Records Management and Data Entry, NYS Tax Department, W A Harriman Campus, Albany NY 12227; telephone 1 800 225-5829. From areas outside the United States and outside Canada, call (518) 485-6800.

Need help?

Internet access: www.nystax.gov (for information, forms, and publications)
Fax-on-demand forms: 1 800 748-3676
Telephone assistance is available from 8:00 A.M. to 5:00 P.M. (eastern time), Monday through Friday. 1 800 698-2931
To order forms and publications: 1 800 462-8100
From areas outside the U.S. and outside Canada: (518) 485-6800
Hearing and speech impaired (telecommunications device for the deaf (TDD) callers only): 1 800 634-2110
Persons with disabilities: In compliance with the Americans with Disabilities Act, we will ensure that our lobbies, offices, meeting rooms, and other facilities are accessible to persons with disabilities. If you have questions about special accommodations for persons with disabilities, please call 1 800 972-1233.

## **CONSULTANT CONTRACTOR FORMS**

## **EMPLOYEE INFORMATION TO BE REPORTED BY CERTAIN CONSULTANT CONTRACTORS**

### **Instructions for Completing Form A and B**

Form A and Form B should be completed for contracts for consulting services in accordance with the following

#### **Form A - Contractor's Planned Employment (to be completed and submitted with bid/quote)**

- **Employment Category:** enter the specific occupation(s), as listed in the O\*NET occupational classification system, which best describe the planned employees to provide services under the contract.

(Note: Access the O\*NET database, which is available through the US Department of Labor's Employment and Training Administration, on-line at [online.onetcenter.org](http://online.onetcenter.org) to find a list of occupations.)

- **Number of Employees:** enter the total number of employees in the employment category to be employed to provide services under the contract including part time employees and employees of subcontractors.
- **Number of hours:** enter the total number of hours to be worked by the employees in the employment category.
- **Amount Payable under the Contract:** enter the total amount payable by the State to the State contractor under the contract, for work by the employees in the employment category.

#### **Form B – Contractor's Annual Employment Report. (to be completed by May 1<sup>st</sup> of each year for each consultant contract in effect at any time between the preceding April 1<sup>st</sup> through March 31<sup>st</sup> fiscal year and submitted to the Department of Civil Service, Office of the State Comptroller and Office of General Services)**

- **Scope of Contract:** choose a general classification of the single category that best fits the predominate nature of the services provided under the contract.
- **Employment Category:** enter the specific occupation(s), as listed in the O\*NET occupational classification system, which best describe the employees providing services under the contract.

(Note: Access the O\*NET database, which is available through the US Department of Labor's Employment and Training Administration, on-line at [online.onetcenter.org](http://online.onetcenter.org) to find a list of occupations.)

- **Number of Employees:** enter the total number of employees in the employment category employed to provide services under the contract during the report period, including part time employees and employees of subcontractors.
- **Number of hours:** enter the total number of hours worked during the report period by the employees in the employment category.
- **Amount Payable under the Contract:** enter the total amount paid by the State to the State contractor under the contract, for work by the employees in the employment category, for services provided during the report period.

<b>OSC Use Only:</b> Reporting Code: Category Code: Date Contract Approved:
--------------------------------------------------------------------------------------

**FORM A**

<b>State Consultant Services - Contractor's Planned Employment          From Contract Start Date Through The End Of The Contract Term</b>
-----------------------------------------------------------------------------------------------------------------------------------------------

State Agency Name:	Agency Code:
Contractor Name:	Contract Number:
Contract Start Date:    /    /	Contract End Date:    /    /

O*Net Employment Category <small>(see O*Net on-line at <a href="http://online.onetcenter.org">online.onetcenter.org</a>)</small>	Number of Employees	Number of hours to be worked	Amount Payable Under the Contract
Total this page	0	0	\$ 0.00
Grand Total			

Name of person who prepared this report:

Title:

Phone #:

Preparer's Signature:

Date Prepared:    /    /

(Use additional pages, if necessary)

Page    of

<p><b>OSC Use Only:</b>                  Reporting Code:                  Category Code:</p>
------------------------------------------------------------------------------------------------------

<p><b>State Consultant Services                  Contractor's Annual Employment Report                  Report Period: April 1,            to March 31,</b></p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Contracting State Agency Name: _____ Agency Code: _____                  Contract Number: _____                  Contract Term:    /    /            to    /    /                  Contractor Name: _____                  Contractor Address: _____                  Description of Services Being Provided: _____</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><b>Scope of Contract (Choose one that best fits):</b>                  Analysis <input type="checkbox"/>    Evaluation <input type="checkbox"/>    Research <input type="checkbox"/>    Training <input type="checkbox"/>                  Data Processing <input type="checkbox"/>    Computer Programming <input type="checkbox"/>    Other IT consulting <input type="checkbox"/>                  Engineering <input type="checkbox"/>    Architect Services <input type="checkbox"/>    Surveying <input type="checkbox"/>    Environmental Services <input type="checkbox"/>                  Health Services <input type="checkbox"/>    Mental Health Services <input type="checkbox"/>                  Accounting <input type="checkbox"/>    Auditing <input type="checkbox"/>    Paralegal <input type="checkbox"/>    Legal <input type="checkbox"/>    Other Consulting <input type="checkbox"/></p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

O*Net Employment Category (see O*Net on-line at <a href="http://online.onetcenter.org">online.onetcenter.org</a> )	Number of Employees	Number of Hours Worked	Amount Payable Under the Contract
<b>Total this page</b>	0	0	\$ 0.00
<b>Grand Total</b>			

<p>Name of person who prepared this report: _____                  Preparer's Signature: _____                  Title: _____ Phone #: _____                  Date Prepared:    /    /</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Use additional pages if necessary)



**NEW YORK STATE OFFICE OF THE STATE COMPTROLLER  
SUBSTITUTE FORM W-9:  
REQUEST FOR TAXPAYER IDENTIFICATION NUMBER & CERTIFICATION**

*TYPE OR PRINT INFORMATION NEATLY. PLEASE REFER TO INSTRUCTIONS FOR MORE INFORMATION.*

**Part I: Vendor Information**

1. Legal Business Name:	2. If you use a DBA, please list below:
3. Entity Type (Check one only): <input type="checkbox"/> Sole Proprietor <input type="checkbox"/> Partnership <input type="checkbox"/> Limited Liability Co. <input type="checkbox"/> Business Corporation <input type="checkbox"/> Unincorporated Association/Business <input type="checkbox"/> Federal Government <input type="checkbox"/> State Government <input type="checkbox"/> Public Authority <input type="checkbox"/> Local Government <input type="checkbox"/> School District <input type="checkbox"/> Fire District <input type="checkbox"/> Other _____	

**Part II: Taxpayer Identification Number (TIN) & Taxpayer Identification Type**

1. Enter your TIN here: <i>(DO NOT USE DASHES)</i>	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="width: 20px; height: 20px;"></td> </tr> </table>										
2. Taxpayer Identification Type (check appropriate box): <input type="checkbox"/> Employer ID No. (EIN) <input type="checkbox"/> Social Security No. (SSN) <input type="checkbox"/> Individual Taxpayer ID No. (ITIN) <input type="checkbox"/> N/A (Non-United States Business Entity)											

**Part III: Address**

1. Physical Address: Number, Street, and Apartment or Suite Number	2. Remittance Address: Number, Street, and Apartment or Suite Number
City, State, and Nine Digit Zip Code or Country	City, State, and Nine Digit Zip Code or Country

**Part IV: Exemption from Backup Withholding and Certification**

For payees exempt from Backup Withholding, check the box below. Valid explanation required for exemption. See Instructions.

**Exempt from Backup Withholding**

The Internal Revenue Service does not require your consent to any provision of this document other than the certifications required to avoid backup withholding. Under penalties of perjury, I certify that the number shown on this form is my correct Taxpayer Identification Number (TIN).

**Sign Here:**

Signature	Date	
Print Preparer's Name	Phone Number	Email Address

**Part V: Contact Information – Individual Authorized to Represent the Vendor**

Vendor Contact Person: \_\_\_\_\_ Title: \_\_\_\_\_

Contact's Email Address: \_\_\_\_\_ Phone Number: \_\_\_\_\_

*DO NOT SUBMIT FORM TO IRS — SUBMIT FORM TO NYS ONLY AS DIRECTED*

**FOR OSC USE ONLY**

## APPENDIX C

# SAMPLE SERVICE AGREEMENT



**ANDREW M. CUOMO**  
GOVERNOR

**ROANN M. DESTITO**  
COMMISSIONER

STATE OF NEW YORK  
**EXECUTIVE DEPARTMENT**  
**OFFICE OF GENERAL SERVICES**  
MAYOR ERASTUS CORNING 2ND TOWER  
THE GOVERNOR NELSON A. ROCKEFELLER EMPIRE STATE PLAZA  
ALBANY, NEW YORK 12242

### **SERVICE AGREEMENT**

**Contract No.** \_\_\_\_\_

**Solicitation #** \_\_\_\_\_

**Term** \_\_\_\_\_

This agreement made this \_\_\_\_ day of \_\_\_\_\_ 2012 by and between THE PEOPLE OF THE STATE OF NEW YORK, hereinafter referred to as "STATE", acting by and through Roann M. Destito, the Commissioner of General Services, thereinafter referred to as "Commissioner", whose office is in the Mayor Erastus Corning II Tower, Empire State Plaza, the City and County of Albany, State of New York and; \_\_\_\_\_, located at \_\_\_\_\_, hereinafter referred to as "Contractor."

WHEREAS, the State has formally requested contractors for proposals for the following described PROJECT: IT Services to Develop a Web-Based Application "eHBITS," a copy of which is hereto annexed and made a part thereof, hereinafter referred to as the "Request for Quote" or "RFQ", and;

WHEREAS, the Commissioner has determined that the contractor is the successful bidder, and the contractor is willing to undertake the services and to provide the necessary materials, labor, and equipment in connection therewith.

NOW THEREFORE, in consideration of the terms hereinafter mentioned and also the covenants and obligations moving to each party hereto from the other, the parties hereto do hereby agree as follows:

- 1) The Contractor agrees to perform this contract and to furnish the services, labor and materials required in connection therewith, in accordance with all of the conditions, covenants and representations contained in the proposal and specifications which are hereto annexed and incorporated herein and which are deemed to be a part of this contract with the same force and effect as herein set forth separately, specifically and at length, and the State agrees to pay to the Contractor therefore those rates specified in the proposal.

- 2) It is understood by and between the parties hereto that this agreement shall be deemed executory to the extent of the monies available to the State and no liability on account thereof shall be incurred by the State beyond monies available for the purpose thereof.
- 3) The established maximum value of this contract shall be set at \$ \_\_\_\_\_; and shall not be exceeded.
- 4) Inconsistencies:

In the event of any discrepancy, disagreement or ambiguity between this contract agreement and its Appendices, or between any Appendices, the documents shall be given preference in the following order to interpret and to resolve such discrepancy, disagreement or ambiguity:

- (i) Appendix A
- (ii) This Contract Agreement
- (iii) Request for Quote including any Addenda
- (iv) The Contractor's Quote

- 5) Termination:

A.) Termination

The Office of General Services may, upon thirty (30) days notice, terminate this Agreement in the event of the Contractor's failure to comply with any of the bid's requirements unless the Contractor obtained a waiver of the requirement. In addition, OGS may also terminate this Agreement upon ten (10) days written notice if the Contractor makes any arrangement or assignment for the benefit of creditors. Furthermore, OGS shall have the right, in its sole discretion, at any time to terminate this Agreement, or any unit portion thereof, with or without cause, by giving thirty (30) days written notice of termination to the Contractor. Any termination by OGS under this Section shall in no event constitute or be deemed a breach of this Agreement and no liability shall be incurred by or arise against the Office of General Services, its agents and employees therefore for lost profits or any other damages.

B.) Procurement Lobbying Termination

The Office of General Services reserves the right to terminate this Agreement in the event it is found that the certification filed by the Contractor in accordance with New York State Finance Law §139-k was intentionally false or intentionally incomplete. Upon such finding, the Office of General Services may exercise its termination right by providing written notification to the Contractor in accordance with the written notification terms of this Agreement.

- 6) By signing this Service Agreement, Contractor certifies that all information provided to OGS with respect to State Finance Law §139-k is complete, true and accurate.
- 7) In accordance with 5 NYCRR §142.13, the Contractor acknowledges that if it is found to have willfully and intentionally failed to comply with the MWBE participation goals set forth in the Contract, such finding constitutes a breach of contract and the Contractor shall be liable to OGS for liquidated or other appropriate damages, as set forth in the solicitation.

IN WITNESS WHEREOF, the parties hereto have executed this agreement as of the day and year first above written.

CONTRACT NO. \_\_\_\_\_

(To be completed by Agency)

**AGENCY CERTIFICATION**

(In addition to the acceptance of this contract, I also certify that original copies of this signature page will be attached to all other exact copies of this contract.)

**AGENCY SIGNATURE**

**CONTRACTOR'S SIGNATURE**

\_\_\_\_\_  
SIGNATURE

\_\_\_\_\_  
SIGNATURE

\_\_\_\_\_  
PRINT NAME

\_\_\_\_\_  
PRINT NAME

\_\_\_\_\_  
TITLE

\_\_\_\_\_  
TITLE

DATED \_\_\_\_\_

DATED \_\_\_\_\_

COMPANY \_\_\_\_\_

\_\_\_\_\_  
ADDRESS

\_\_\_\_\_  
CITY STATE / ZIP

\_\_\_\_\_  
TELEPHONE NUMBER

\_\_\_\_\_  
FEDERAL I.D. NUMBER

\_\_\_\_\_  
NYS VENDOR I.D. NUMBER

**ATTORNEY GENERAL'S SIGNATURE**

**COMPTROLLER'S SIGNATURE**

Approved:

\_\_\_\_\_

\_\_\_\_\_

DATED \_\_\_\_\_

DATED \_\_\_\_\_

**INDIVIDUAL, CORPORATION, PARTNERSHIP, OR LLC ACKNOWLEDGMENT**

STATE OF \_\_\_\_\_ }  
: SS.:  
COUNTY OF \_\_\_\_\_ }

On the \_\_\_\_ day of \_\_\_\_\_ in the year 20 \_\_ , before me personally appeared \_\_\_\_\_, known to me to be the person who executed the foregoing instrument, who, being duly sworn by me did depose and say that \_he resides at \_\_\_\_\_, Town of \_\_\_\_\_, County of \_\_\_\_\_, State of \_\_\_\_\_; and further that:

**[Check One]**

**If an individual):** \_he executed the foregoing instrument in his/her name and on his/her own behalf.

**If a corporation):** \_he is the \_\_\_\_\_ of \_\_\_\_\_, the corporation described in said instrument; that, by authority of the Board of Directors of said corporation, \_he is authorized to execute the foregoing instrument on behalf of the corporation for purposes set forth therein; and that, pursuant to that authority, \_he executed the foregoing instrument in the name of and on behalf of said corporation as the act and deed of said corporation.

**If a partnership):** \_he is the \_\_\_\_\_ of \_\_\_\_\_, the partnership described in said instrument; that, by the terms of said partnership, \_he is authorized to execute the foregoing instrument on behalf of the partnership for purposes set forth therein; and that, pursuant to that authority, \_he executed the foregoing instrument in the name of and on behalf of said partnership as the act and deed of said partnership.

**If a limited liability company):** \_he is a duly authorized member of \_\_\_\_\_, LLC, the limited liability company described in said instrument; that \_he is authorized to execute the foregoing instrument on behalf of the limited liability company for purposes set forth therein; and that, pursuant to that authority, \_he executed the foregoing instrument in the name of and on behalf of said limited liability company as the act and deed of said limited liability company.

\_\_\_\_\_  
**Notary Public**  
**Registration No.** \_\_\_\_\_

**State of:** \_\_\_\_\_

**EXHIBIT 1**

**HBITS PROCESS FLOW AND SELECTION FORMS**

**“eHBITS” RFQ 1768**

**July 2, 2012**



**AGENCY**

**MSP (OGS)**

**CONTRACTOR**

Agency completes a **Task Order (TO) Request Form 1**

**Total Process Time: 23-25 business days**  
(All timeframes are in business days)

OGS reviews the request, validates all data fields and compliance with contractual terms, and posts on the **Contractor Bulletin Board** (for contractors only). (2-3 Days)

Contractors receive an automatic e-mail that a TO is posted. Contractors propose candidates using **Form 2**. (10 days/5 days expedited with HBITS Group pre-approval)  
\*Does not include day of transmission to vendors.

Agency reviews the forms using **Form 3A** to determine the most technically qualified candidates to interview. Notifies OGS through portal of selected candidates for interview; all other candidates are released. Step by Step Procedures are below:  
Part 1- 5 days  
• Form 3A is completed with initial score.  
Part 2- 5 days  
• Agency schedules interviews (min. of 3 req.) but agency may interview more  
• Interviews to validate initial score and qualifications  
• Completes Evaluation Form 3B on all interviewees  
• Selects the highest technically scored candidate after interview  
• Updates the TO in the portal, releases remaining non-selected candidates  
• Work with Contractor to start the consultant

OGS reviews the responses, validates mandatory requirements from Contract are met, and releases appropriate responses to the Agency. (1-2 days)



OGS ensures the Contractor and the Agency have completed the process the forms in the database.

Contractor has **10 business days** to perform all Candidate placement requirements and finalizes onboarding of consultants.

OGS ensures the job has begun.

**eHIBTS RFQ -- Above this line**

Agency can complete a **TO Modification Request Form 5**  
Agency can complete an **Issue Form 6**

OGS would approve and oversee either form

Contractor can complete a **TO Modification Request Form 5**  
Contractor can complete an **Issue Form 7**

**Future Portal Development**

**Future Portal Development**

HBITS Group receives the invoice and Form 8 from each contractor; validates both against the TOs and Form 8. If a discrepancy is found, HBITS group checks with agency and contractor. If error, adjusts next month's payment.

• Contractor sends an invoice to OGS Finance.  
• Contractor completes a **Monthly Report Form 8** and sends to the HBITS group for verification-off

**Job Completion –**  
Agency completes **TO Satisfaction Form 4** and submits to HBITS Database

Annually –  
OGS completes **Contractor Performance Evaluation Form 9** and amends Contractor Lists

**NON-EXECUTIVE AGENCY**

**MSP (OGS)**

**CONTRACTOR**



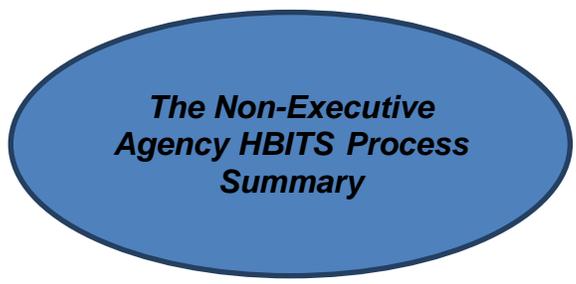
Non-Executive Agency completes a **Task Order (TO) Form 1**

OGS posts on the **Contractor Bulletin Board** (for contractors only) (**2-3 Days**)

Contractors receive an automatic e-mail that a TO is posted. Contractors propose candidates using **Form 2. (10 Days)**  
\*Does not include day of transmission to vendors.

**Total Process Time: 23-25 business days**

Non-Executive Agency reviews the bid forms using **Form 3A** to determine the most technically qualified candidates to interview. Select the vendors for interview; all other candidates are released. Step by Step Procedures are below:  
Part 1- 5 Days  
• Form 3A is completed with initial score.  
Part 2- 5 Days  
• Schedules interviews (min. of 3 req.) but agency may interview more  
• Interviews to validate initial score and qualifications  
• Completes Evaluation Form 3B on all interviewees  
• Selects the highest technically scored candidate after interview  
• Updates the TO in the portal, releases remaining non-selected candidates  
  
Non-Executive Agency works with Contractor to start the consultant.



Contractor has **10 business days** to perform all Candidate placement requirements and finalizes onboarding of consultants.

**eHIBTS RFQ -- Above this line**

Non-Agency can complete a TO Modification Request **Form 5**

Contractor can complete a TO Modification Request **Form 5**

**Future Portal Development**

Non-Agency can complete an **Issue Form**

OGS would oversee either form

Contractor can complete an **Issue Form**

**Future Portal Development**

Non-Executive Agency verifies and pays the invoice

Contractor bills the Non-Executive Agency and sends the invoice

**Job Completion** – Agency completes **TO Satisfaction Form 9** and submits to HBITS Database

**Annually** – OGS completes **Contractor Performance Evaluation Form 9** and amends Contractor lists



## HBITS Form 1:Task Order Request Form

### This Form Will Act as the Main Input into corresponding evaluation forms

A maximum of five (5) candidates may be requested if all positions are for the same service group, job title, skill level/demand and position qualifications. Accordingly, this form may change based on final development needs and identified fields below may be adjusted based on the authorized user and may not be applicable to all requests. **All fields will be required or the portal cannot accept the submission.**

Request Date:	Calendar Icon
Agency:	
Agency Contact Name:	
Agency Contact E-mail:	
Agency Contact Phone #:	

Has Agency received the necessary internal agency (management) approvals to support this Task Order Request (if applicable)?	Drop Down (Yes or No)
Has Agency received PTP approval from OITS to support this Task Order Request (if applicable)?	Drop Down (Yes or No)
Has Agency received DOB approval to support this Task Order request (if applicable)?	Drop Down (Yes or No)
Will Federal Funding be used to pay (in part or in full) for this position(s) (if applicable)?	Drop Down (Yes or No)
Is this a request for the NYS Department of Labor? See Section 5.23 of the Contract for additional details.	Drop Down (Yes or No)

Is this a Project or Program specific request (e.g., Connections, SFS)?	Drop Down (Yes or No)
If yes, please list the Project or Program Name:	Text Field
Please provide a short description of the project:	Text Field
Please provide a full listing of the day to day tasks to be performed by the Consultant (be descriptive and specific):	<b>Prefer data entry method to be in bulleted format</b>

Is this an expedited request (within 5 days)? <i>If Yes, please provide detailed justification in the box below. Expedited requests require written pre-approval and are at the sole discretion of the MSP. These requests are not and will not be commonplace. Expedited requests are not available for Non-MSP Users.</i>	Drop Down (Yes Or No) Must have the ability for OGS to modify this field after the Authorized User has submitted the field via the portal
Justification:	(Text Field)
Is this a New or Incumbent request?	Drop Down (New or Incumbent)
Which Service Group is required?	Drop Down (1 or 2)
What is the number of staff being requested? (maximum of 5)	Drop Down (1, 2, 3, 4 or 5)
Which Job Title Category is required?	Drop Down that is linked to the job titles associated with the corresponding service group
Which Skill Level is required?	Drop Down (Junior, Mid-Level, Senior, Expert)
Which Skill Demand is required?	Drop Down (Normal or High)
How many Candidate Response Forms are being requested per Contractor per position?	Drop Down (1 or 2)
When is the Target Start Date? (auto rule of 30 Business Days minimum from date of request)	Calendar Icon with auto rule
How long is the engagement?	Numerical Field with Number of Months
When is the estimated completion date?	Numerical Field with Number of Months

Is this a Full or Part-Time Position (Full time is considered 40 Hours Per Week)?	Drop Down (Full-time or Part Time)		
If Part-Time, enter approximate number of hours per week:	Numerical field		
What are the daily work hours? (note if negotiable or list preferred start and end time):	Text Field		
Where is the Home Base Region? (Add link to Appendix E Region Definitions)	Drop Down (1, 2 or 3)		
Where is the work office located? Enter Building Name (if known) Full Street Address, City, and ZIP	Text Field		
What type of software is typically used by the Agency? (e.g., Agency is an "IBM shop.")	Text Field		
What type of hardware is typically used by the Agency? (e.g., Agency is a "Unisys shop.")	Text Field		
<b>Position Mandatory Qualifications (All Qualifications and points assigned must carry over to the evaluation forms (3A and 3B) and linked to the task order number)</b>			
(Insert Text From Contract for Position Title and Skill Level)		Pass/Fail	
<u>"Mandatory" Qualifications cannot be changed nor can additional "mandatory" qualifications be added</u>		<i>The number of qualifications requested, and the number of points assigned must total 80 points.</i>	
Qualification Number	Requested Qualifications	Points Assigned for Meeting Qualifications  (Always 75% of Max Points) Auto Perform 75% Calculation	Maximum Points Allowed for Exceeding Qualifications (80)
1.	Sample: X Months of Experience in Y doing Z	7.5	10
2.	Sample: X Months of Experience in Y doing Z	7.5	10
3.	Sample: X Months of Experience in Y doing Z	7.5	10
4.	Sample: X Months of Experience in Y doing Z	7.5	10
5.	Sample: X Months of Experience in Y doing Z	7.5	10
6.	Sample: X Months of Experience in Y doing Z	7.5	10
7.	Sample: X Months of Experience in Y doing Z	3.75	5
8.	Sample: X Months of Experience in Y doing Z	3.75	5
9.	Sample: Bachelor's Degree	3.75	5
10.	Sample: Project Management Professional Certification	3.75	5
<b>Requested Qualifications Must Always Total 80 Points For Maximum Score</b>		60 (Required total)	80 (Required total)
<b>Interview Must Always Total (Required Field that cannot be changed)</b>		20	
<b>Total Score (Required Field and Control to be instituted to ensure qualifications total 100)</b>		100	

**Additional Information Requests:**

<b>Are there additional security requirements for the Authorized User?</b>	<b>Drop Down (Yes or No)</b>
<b>Will additional training possibly be required during the Engagement?</b>	<b>Drop Down (Yes or No)</b>
<b>If Yes, provide description of anticipated training.</b>	<b>Text Field</b>
<b>What type, or manner, of Knowledge Transfer is requested during the engagement?</b>	<b>Text Field</b>
<b>Is travel anticipated during the Engagement?</b>	<b>Drop Down (Yes or No)</b>
<b>If Yes, please list anticipated frequency and locations for travel:</b>	<b>Text Field</b>

## HBITS Form 2: Candidate Response Form

Date:	Calendar Icon			
Task Order #:	(Entry in this field will populate several other fields in this form)			
Contractor Name:	Drop Down with 25 Vendor Names			
Should an individual other than the Contract Administrator/ Secondary Contact be contacted about this Candidate?	Drop Down (Yes Or No)			
If Yes, please provide the Contact Name for this Response:	Text field			
Contractor Phone #:	Phone number format for field			
Contractor E-mail:				
Is the proposed Candidate a US Citizen?	Drop Down (Yes or No)			
If Yes, please list the full first name as depicted on the State Driver's license or governmental identification:	Linked to answer above			
If Yes, please list the full last name as depicted on the Driver's license or governmental identification:	Linked to answer above			
If No, please list the full first name as depicted on the Visa/ Passport. *No abbreviations or other derivations are allowed.	Linked to answer above			
If No, please list the full last name as depicted on the Visa/ Passport. *No abbreviations or other derivations are allowed.	Linked to answer above			
Visa #	Linked to answer above			
Type of Visa (e.g., L-1, H1-B)	Linked to answer above			
Candidate Full Name:	Combination of answer to first and last name questions			
Was a pre-interview of the conducted in accordance with Section 6.3.4 of the HBITS Contract? <i>*Failure to Perform Pre-Interview of Candidate will result in automatic rejection of Candidate.</i>	Drop Down (Yes or No) Answer of No will not allow user to submit candidate via web portal			
If Yes, what date was the Pre-Interview conducted?	Calendar Icon			
Does the Candidate anticipate any absences during the engagement (See Section 6.3.8 of the HBITS contract)?	Drop Down (Yes or No)			
If Yes, please list the start and end dates of each absence?	Text Field			
Proposed Interview dates (Cannot be earlier than 7 business days after due date of requisition)	Text Field			
Project:	Pre-Populated based on Task Order Number			
Service Group :	Pre-Populated based on Task Order Number			
Job Title(s) Category:	Pre-Populated based on Task Order Number			
Skill Level:	Pre-Populated based on Task Order Number			
Skill Demand Required:	Pre-Populated based on Task Order Number			
Home Based Region:	Pre-Populated based on Task Order Number			
Employment Status of Consultant:	Required Field (Drop Down of Direct Employee, Subcontracted Employee or Independent Contractor)			
<b>Identify any and all subcontractors involved with the placement of the Candidate below:</b>				
Name	Address	Is subcontractor an M/WBE? (link to ESDC database)	Is subcontractor an SBE? (link to show definition of SBE)	Is this subcontractor paying Hourly Wage Rate for Candidate?
				Drop Down (Yes or No)
				Drop Down (Yes or No)

	<b>ALL QUALIFICATIONS WILL BE LINKED TO THE TASK ORDER NUMBER. AREAS THAT REQUIRE A DETAILED DESCRIPTION WILL BE MANUALLY ENTERED BY CONTRACTORS WHEN SUBMITTING RESPONSE VIA PORTAL</b>	
<b>Mandatory Qualification</b>	Pre-Defined: (Insert Text From Contract for Position Title and Skill Level)	Pass/Fail
	Provide a detailed description of how the proposed Candidate meets the mandatory qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank (Vendors Do Not Score)
<b>Qualification Number</b>	<b>Requested Qualifications:</b>	<b>Max Points Assigned</b>
1.	Sample: X Months of Experience in Y doing Z	10
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
2.	Sample: X Months of Experience in Y doing Z	10
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
3.	Sample: X Months of Experience in Y doing Z	10
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
4.	Sample: X Months of Experience in Y doing Z	10
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
5.	Sample: X Months of Experience in Y doing Z	10
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
6.	Sample: X Months of Experience in Y doing Z	10
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
7.	Sample: X Months of Experience in Y doing Z	5
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
8.	Sample: X Months of Experience in Y doing Z	5
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
9.	Sample: Bachelor's Degree	5
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
10.	Sample: Project Management Professional (PMP)	5
	Provide a detailed description of how the proposed Candidate meets the requested qualification. Include name(s) of previous employer(s), start and end dates of engagement(s), reference, and any additional applicable information.	Leave This Cell Blank
<b>Requested Qualifications Must Total</b>		<b>80</b>

<b>Interview Must Total</b>				<b>20</b>
<b>Total Score</b>				<b>100</b>
<b>REFERENCES (Optional Fields)</b>				
	<u>Name</u>	<u>Company</u>	<u>Phone</u>	<u>E-Mail</u>
Reference #1 (Optional)				
Reference #2 (Optional)				
Reference #3 (Optional)				

<b>Additional Information Requests: Required Fields</b>	
Can the proposed Candidate(s) meet the additional Security Requirements requested (Note All That Apply)?	Drop Down (Yes or No)
If Other is Selected, please provide relevant information:	Drop Down (Yes or No)
Can the proposed Candidate(s) meet the additional training possibly required during the Engagement?	Drop Down (Yes or No)
Can the Candidate meet the request type and manner of knowledge transfer requested during the engagement?	Drop Down (Yes or No)
Can the Candidate(s) meet the travel anticipated during the Engagement?	Drop Down (Yes or No)

\*Resumes can be attached, in addition to the completed form.

\*\*If education credentials were requested, Contractor must attached proof of degree equivalency as required by Section 6.3.3 of the HBITS contract

\*\*\* The Contractor's agreement to comply with the provisions of this form is a material representation of fact upon which reliance was placed when the Authorized User determined to enter into an engagement with the Contractor.

**HBITS Form 3A:**  
**Authorized User Preliminary Technical Evaluation Form**

Candidate Specific Form: Please complete this form for each Candidate Response Form reviewed.

<b>Date:</b>	Calendar Icon
<b>Task Order #:</b>	Entry of this number will pre-populate several fields
<b>Contractor Name:</b>	Drop Down of vendors associated with submissions to individual task order numbers
<b>Candidate Full Name</b>	Drop Down of candidate names associated with submissions to individual task order numbers
<b>List any anticipated absences and the dates of such absences the Candidate may have during the engagement:</b>	Linked to Form 2 based on the Contractors submission for the candidate submitted to the individual task order number
	Pre-Populated based on the task order number
<b>Project:</b>	Pre-Populated based on the task order number
<b>Service Group :</b>	Pre-Populated based on the task order number
<b>Job Title(s) Category:</b>	Pre-Populated based on the task order number
<b>Skill Level:</b>	Pre-Populated based on the task order number
<b>Skill Demand Required:</b>	Pre-Populated based on the task order number
<b>Home Base Region</b>	Pre-Populated based on the task order number
<b>Additional Security Requirements for the Authorized User:</b>	Pre-Populated based on the task order number
<b>If Other is listed, please provide relevant information:</b>	Pre-Populated based on the task order number
<b>Additional Training Possibly Required During Engagement?</b>	Pre-Populated based on the task order number
<b>If Yes, Provide description of anticipated training?</b>	Pre-Populated based on the task order number
<b>Type or Manner of Knowledge Transfer requested during the Engagement:</b>	Pre-Populated based on the task order number
<b>Is Travel anticipated during the Engagement?</b>	Pre-Populated based on the task order number
<b>If Yes, please list anticipated frequency and locations for travel:</b>	Pre-Populated based on the task order number

**CONTINUED ON NEXT PAGE**

**Scoring Criteria**

- 1) If Candidate Does Not Pass Mandatory Qualification – Score “Fail,” Enter Rationale in Comments Column and discontinue evaluation
- 2) Candidate Meets Qualifications- Assign Score of 75% of Max Points
- 3) Candidate Exceeds Requested Qualification – Score Full Points Assigned, Enter Information in Comments Column Detailing Rationale for Exceeding Qualifications
- 4) Candidate Does Not Meet Requested Qualification – Score 0 Points. No Partial Points Allowed for Not Meeting Requested Qualifications

**Mandatory Qualification**

	The field below is pre-populated based on the initial Form 1 submission by the Authorized User	<u>Task Order Points</u>	<u>Candidate Score</u>	<u>Comments</u>
1.	Pre-Defined: (Insert Text From Contract for Position Title and Skill Level)	Pass/Fail	Cannot Exceed Max Points	Text Field

**Requested Qualifications**

<u>Qualification Number</u>	<u>Qualifications below are pre-populated based on the initial Form 1 submission by the Authorized user</u>	<u>Points Assigned for Meeting Qualifications (Always 75% of Max Points)</u>	<u>Points Assigned for Exceeding Qualifications (Max Points)</u>	<u>Candidate Score</u>	<u>Comments</u>
1.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
2.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
3.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
4.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
5.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
6.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
7.	Sample: X Months of Experience in Y doing Z	3.75	5	Cannot Exceed Max Points	Text Field
8.	Sample: X Months of Experience in Y doing Z	3.75	5	Cannot Exceed Max Points	Text Field
9.	Sample: Bachelor’s Degree	3.75	5	Cannot Exceed Max Points	Text Field
10.	Sample: Project Management Professional (PMP) Certification	3.75	5	Cannot Exceed Max Points	Text Field
<b>Total Points:</b>		60	80		

# HBITS Preliminary Technical Evaluation Summary Sheet

Summary Sheet: After completion of all the preliminary technical evaluations, please complete this form summarizing the results.

<u>Task Order Number</u>	<u>Entry of this field will populate all candidate names, contractor names and the scores associated.</u>	<u>Date Of Completion</u>	<u>Calendar Icon</u>
<b>Candidates Selected for Interviews (minimum of three, maximum of 10 per position)</b> <b>*Must Interview Three (3) Highest Scoring Candidates Before Selecting a Lower Ranked Candidates</b>			
<u>Candidate Ranking</u>	<u>Candidate Full Name</u>	<u>Contractor Name</u>	<u>Total Score</u>
1.			Drop Down (Yes/No)
2.			Drop Down (Yes/No)
3.			Drop Down (Yes/No)
4.			Drop Down (Yes/No)
5.			Drop Down (Yes/No)
6.			Drop Down (Yes/No)
7.			Drop Down (Yes/No)
8.			Drop Down (Yes/No)
9.			Drop Down (Yes/No)
10.			Drop Down (Yes/No)
11.			Drop Down (Yes/No)
12.			Drop Down (Yes/No)
13.			Drop Down (Yes/No)
14.			Drop Down (Yes/No)
15.			Drop Down (Yes/No)
16.			Drop Down (Yes/No)
17.			Drop Down (Yes/No)
18.			Drop Down (Yes/No)
19.			Drop Down (Yes/No)
20.			Drop Down (Yes/No)

**Selection of “No” for any of the candidates above will release that candidate from the task order and allow the Contractor to resubmit that candidate for other Task Order requests. An email notification will be sent to all Contractors whose candidates were not identified for interviews.**

**HBITS Form #3B:**  
**Authorized User Interview Evaluation Form**

Candidate Specific Form: Please complete this form for each Candidate interview conducted.

<b>Date:</b>	Calendar Icon
<b>Task Order #:</b>	Entry of this number will populate all but two fields on this page
<b>Contractor Name:</b>	Pre-Populated and linked to Task Order Number
<b>Candidate Full Name</b>	Web Portal User will select a candidate that is linked to the Task Order Number
<b>List any anticipated absences and the dates of such absences the Consultant may have during the engagement:</b>	Pre-Populated and linked to Task Order Number
	Pre-Populated and linked to Task Order Number
<b>Project:</b>	Pre-Populated and linked to Task Order Number
<b>Service Group :</b>	Pre-Populated and linked to Task Order Number
<b>Job Title(s) Category:</b>	Pre-Populated and linked to Task Order Number
<b>Skill Level:</b>	Pre-Populated and linked to Task Order Number
<b>Skill Demand Required:</b>	Pre-Populated and linked to Task Order Number
<b>Home Base Region:</b>	Pre-Populated and linked to Task Order Number
<b>Additional security requirements for the Authorized User:</b>	Pre-Populated and linked to Task Order Number
<b>If Other is listed, please provide relevant information:</b>	Pre-Populated and linked to Task Order Number
<b>Additional training possibly required during Engagement?</b>	Pre-Populated and linked to Task Order Number
<b>If Yes, provide description of anticipated training?</b>	Pre-Populated and linked to Task Order Number
<b>Type or manner of Knowledge Transfer requested during the Engagement:</b>	Pre-Populated and linked to Task Order Number
<b>Is travel anticipated during the Engagement?</b>	Pre-Populated and linked to Task Order Number
<b>If Yes, please list anticipated frequency and locations for travel:</b>	Pre-Populated and linked to Task Order Number
<b>Did interviewee(s) change the interview date and time more than two times OR did not show up?</b>	<b>Drop Down (Yes or No)</b>

**CONTINUED ON NEXT PAGE**

Scoring Criteria

- 1) If Candidate Does Not Pass Mandatory Qualification – Score “Fail,” Enter Rationale in Comments Column and discontinue evaluation
- 2) Candidate Meets Qualifications- Assign Score of 75% of Max Points
- 3) Candidate Exceeds Requested Qualification – Score Full Points Assigned, Enter Information in Comments Column Detailing Rationale for Exceeding Qualifications
- 4) Candidate Does Not Meet Requested Qualification – Score 0 Points. No Partial Points Allowed for Not Meeting Requested Qualifications

**Mandatory Qualifications**

Insert Text From Contract for Position Title and Skill Level)		Task Order Points:			
		Pass/Fail			
Qualification Number	Mandatory Qualifications	Points Assigned for Meeting Qualifications (Always 75% of Max Points)	Points Assigned for Exceeding Qualifications (Max Points)	Candidate Score	Comments and/or Score Change Rationale
1.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
2.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
3.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
4.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
5.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
6.	Sample: X Months of Experience in Y doing Z	7.5	10	Cannot Exceed Max Points	Text Field
7.	Sample: X Months of Experience in Y doing Z	3.75	5	Cannot Exceed Max Points	Text Field
8.	Sample: X Months of Experience in Y doing Z	3.75	5	Cannot Exceed Max Points	Text Field
9.	Sample: Bachelor's Degree	3.75	5	Cannot Exceed Max Points	Text Field
10.	Sample: Project Management Professional (PMP)	3.75	5	Cannot Exceed Max Points	Text Field
<b>Interview Score</b>		20	20		Text Field
<b>Total Points:</b>		<b>80</b>	<b>100</b>		

**Interview Score Criteria: (20 Total Points)**

Part 1: Interview (15 Points)

Highly recommend (15): Candidate provided excellent responses to all interview questions. Candidate has firm grasp on the needs of the agency and appears to have the requisite skill set to successfully perform the duties of the position beyond the expectations of the Authorized User. Candidate should seamlessly fit within the office(s) and work environment of the Authorized User.

Recommend (10): Candidate provided satisfactory answers to all interview questions. Candidate understands the needs of the agency and would satisfactorily complete all tasks required of the candidate.

Do not recommend (0): Candidate was ill-prepared for interview; or Candidate's experience was overstated in the Form 3A submission; or did not understand the needs of the Authorized User; or failed to answer basic questions regarding the skills and experience required for this position.

Part 2: Communication Skills (5 Points)

Excellent (5): The interviewer could clearly understand the information provided by the Candidate without prompting or follow-up. The Candidate was able to clearly communicate his/her skills and experience in response to the questions posed by the interviewer. If selected, the Candidate will be able to effectively speak and interact with staff without assistance.

Average (3): The interviewer could understand the candidate; however, Candidate required a few instances of prompting. The Candidate was mostly able to communicate his/her skills and experience in response to the questions by the interviewer. If selected, the Candidate will be able to speak and interact with staff with minor, if any, assistance.

Poor (0): The interviewer could not understand the Candidate. The Candidate was unable to effectively communicate his/her skills. If selected, the Candidate would have difficulty speaking and interacting with staff without assistance from others.

Additional Justification (please complete if necessary): (Must accommodate text entry)

**\*\*The completion of this information by the Authorized User is optional and it will not be scored.**

**\*\*Not a required field**

<b><i>Is Candidate able to work the estimated work hours?</i></b>	Drop Down (Yes or No)
<b><i>Is Candidate available for the duration of the Task Order?</i></b>	Drop Down (Yes or No)
<b><i>Is Candidate available on the Target Start Date?</i></b>	Drop Down (Yes or No)
<b><i>Is Candidate available to work at the job location?</i></b>	Drop Down (Yes or No)
<b><i>Was the Candidate aware of the Engagement location, daily work hours and job duration prior to the interview?</i></b>	Drop Down (Yes or No)

## HBITS Post-Interview Summary Score Sheet

Summary Sheet: After completion of all the Candidate interviews, please complete this form summarizing the results. This will show as a report on the eHBITS web portal.

<u>Task Order Number</u>	<u>Entry of this field will populate all candidate names, contractor names and the scores associated.</u>	<u>Date:</u>	Calendar Icon
<u>Candidate Ranking</u>	<u>Candidate Full Name</u>	<u>Contractor Name</u>	<u>Total Score</u>
			<u>Candidate Selected for Position?</u>
1.			Drop Down (Yes Or No)
2.			Drop Down (Yes Or No)
3.			Drop Down (Yes Or No)
4.			Drop Down (Yes Or No)
5.			Drop Down (Yes Or No)
6.			Drop Down (Yes Or No)
7.			Drop Down (Yes Or No)
8.			Drop Down (Yes Or No)
9.			Drop Down (Yes Or No)
10.			Drop Down (Yes Or No)
11.			Drop Down (Yes Or No)
12.			Drop Down (Yes Or No)
13.			Drop Down (Yes Or No)
14.			Drop Down (Yes Or No)
15.			Drop Down (Yes Or No)
16.			Drop Down (Yes Or No)
17.			Drop Down (Yes Or No)
18.			Drop Down (Yes Or No)
19.			Drop Down (Yes Or No)
20.			Drop Down (Yes Or No)

**Selection of “No” for any of the candidates above will release that candidate from the task order and allow the Contractor to resubmit that candidate for other Task Order requests. An email notification will be sent to all Contractors whose candidates were not identified for interviews. Email notification also sent to Contractor(s) whose candidate were selected.**

## Exhibit 2



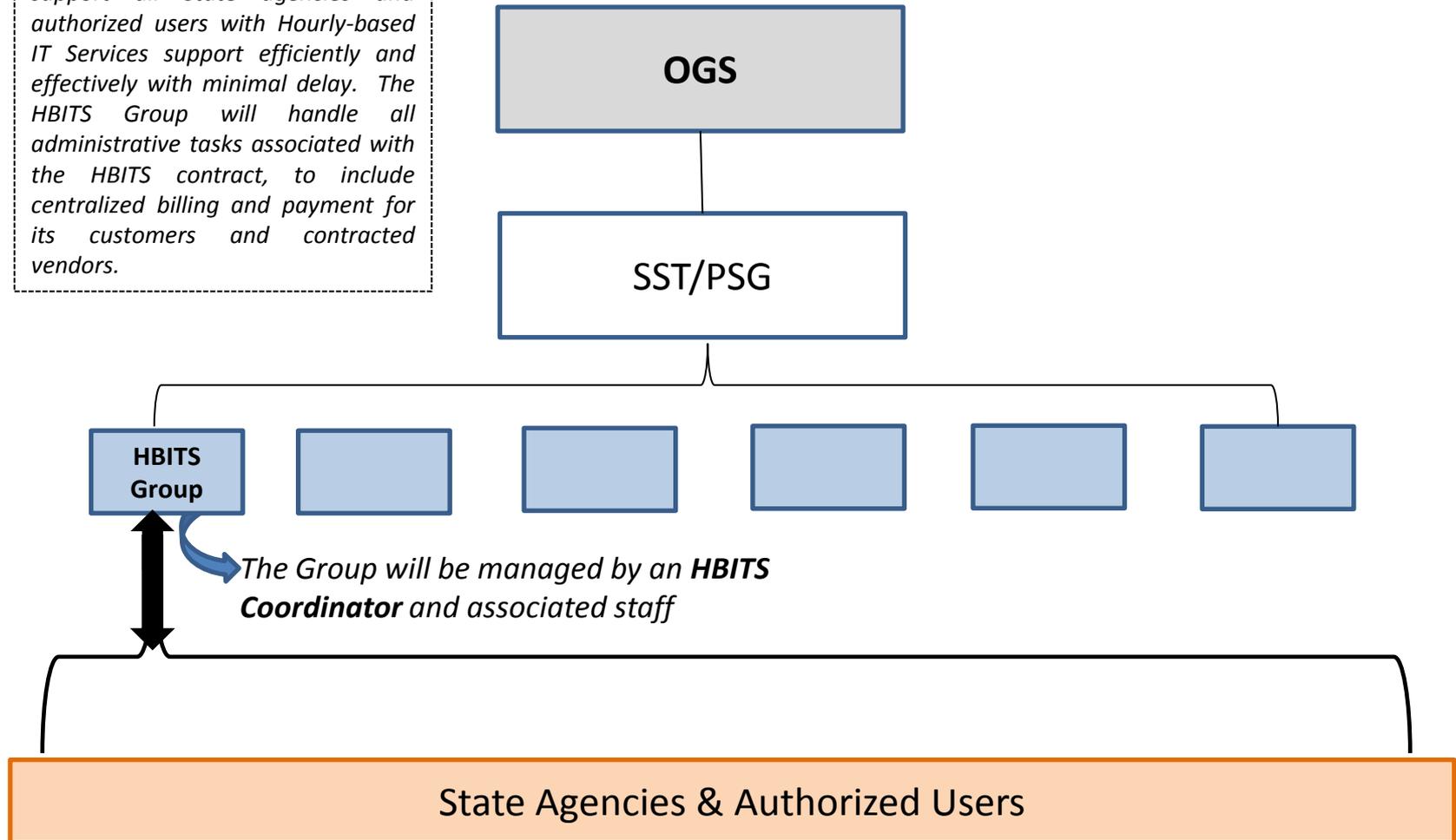
# New York State Hourly Based IT Staff Services (HBITS)

## Portal and Database Development “eHBITS”

July 2, 2012

# HBITS Group

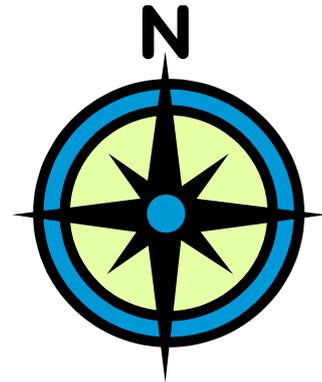
**Mission:** The **HBITS Group** will support all State agencies and authorized users with Hourly-based IT Services support efficiently and effectively with minimal delay. The HBITS Group will handle all administrative tasks associated with the HBITS contract, to include centralized billing and payment for its customers and contracted vendors.



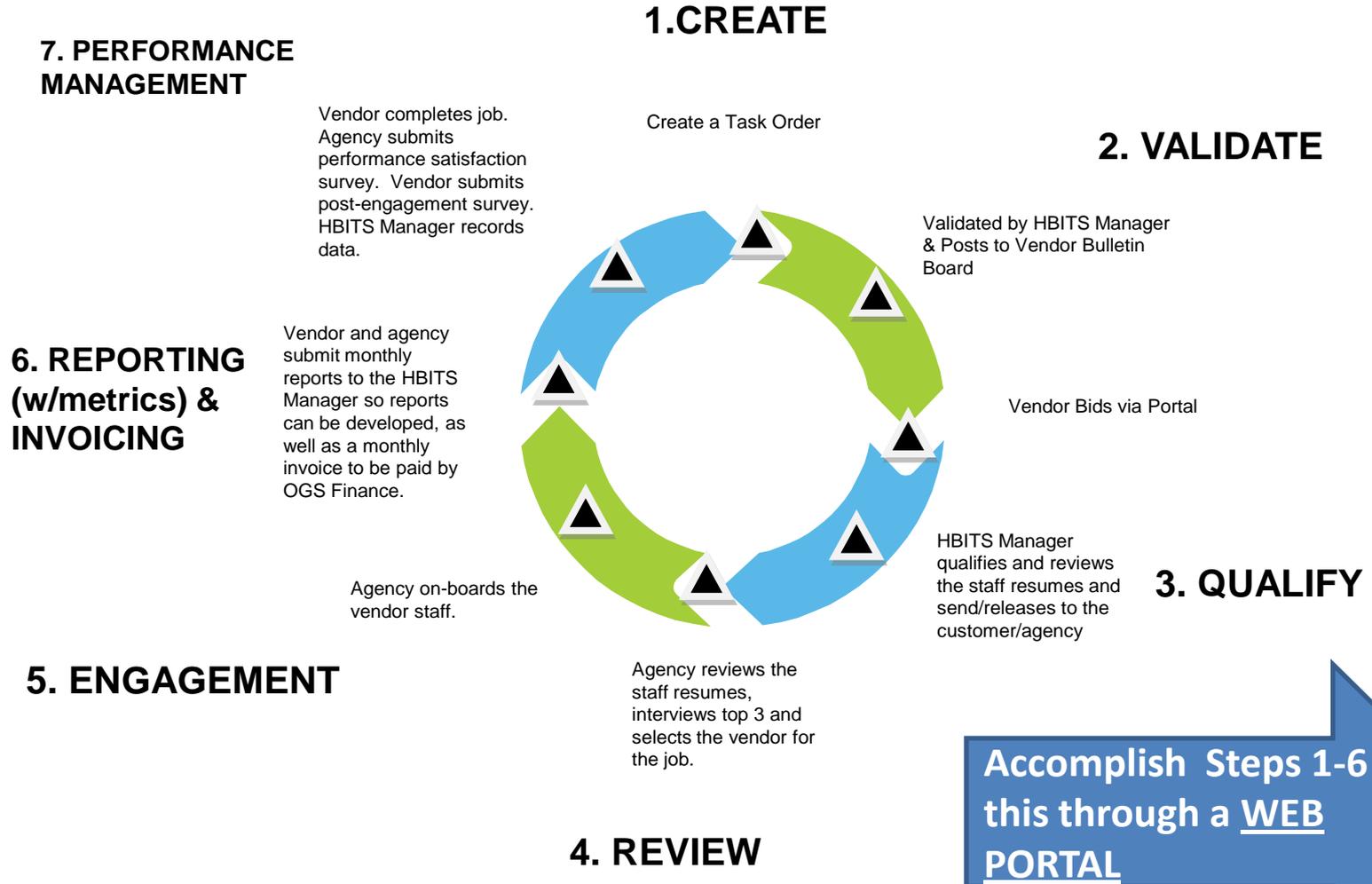
# HBITS Workflow

***OGS has developed a process and workflow that is:***

- Consistent
- Efficient
- Has Flexible Reporting Capability
- Affordable and non-proprietary
- Includes Performance Management Metrics
- Results in Time Savings for the user and management staff
- Leaves Decision-making and oversight of the vendor at the Agency/Customer level
- Includes oversight, but not micromanaged
- Is Web-Based; can be accessed at any time and anywhere and used to communicate with all users
- Results in Good, Solid Matchmaking for agency/customer and vendor



# The HBITS Workflow

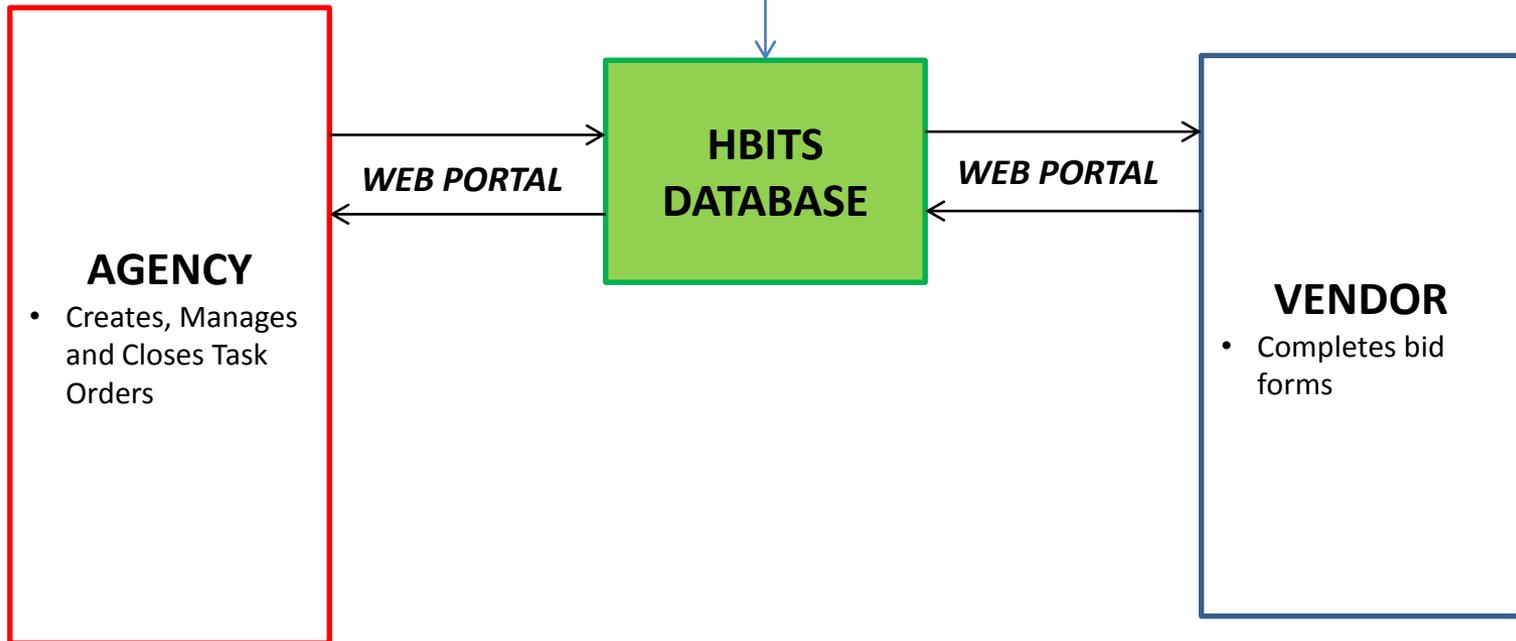


# “eHBITS” – Portal and Database

This is what OGS is seeking an outside vendor to develop.

**HBITS Group  
Coordinator  
& Staff**

- Evaluates
- Validates
- Manages



# eHBITS Web Portal

*The following pages are for illustrative purposes only. There will be modifications to the information depicted on each proposed site (page) via the final portal solution.*



Governor Andrew M. Cuomo

# OFFICE OF GENERAL SERVICES



Commissioner RoAnn M. Destito

[BUILDING ADMINISTRATION](#) | [REAL ESTATE SERVICES](#) | [DESIGN & CONSTRUCTION](#) | [CENTRALIZED PROCUREMENT](#) | [ADMINISTRATION / SUPPORT](#)

ABOUT OGS

CONTACT US

PRESS RELEASES

FREEDOM OF INFORMATION

CORE SERVICES FOR

[STATE & LOCAL GOVERNMENT](#)

[BUSINESS COMMUNITY](#)

[SCHOOLS AND NONPROFITS](#)

[THE GENERAL PUBLIC](#)

MOST REQUESTED LINKS

- [Centralized Procurement Web Portal](#)
- [Minority/Women Owned Businesses](#)
- [Design & Construction Opportunities](#)
- [Documents For Open Meetings](#)
- [Executive Order 4](#)
- [Strategic Sourcing](#)
- [Parking For State Employees](#)
- [State Surplus Property & Auctions](#)
- [Surplus Real Property Sales](#)
- [+ More](#)

© 2012 NEW YORK STATE OGS

[Privacy Policy](#) | [Website Disclaimer](#)

[Accessibility](#) | [Rules & Regulations](#)

## WHAT'S NEW FOR 04/04/2012



**OPEN FOR BUSINESS**  
**NYSStore.com**

**OVER 450 VEHICLES FOR SALE ON eBay**

**Dunn Memorial Bridge Construction to Begin**  
[Learn More](#)

**NEW YORK STATE SURPLUS REAL PROPERTY**  
New York's Real Estate Website



Stay Connected with Events at the Plaza

## VISITING

THE EMPIRE STATE PLAZA AND STATE CAPITOL

[PLACES TO SEE](#)

[PUBLIC EVENTS](#)

[DIRECTIONS](#)

[PARKING](#)

[DISABLED ACCESS](#)

[DINING](#)



Search

Entire Site  Procurement

GO



GOVERNOR ANDREW M. CUOMO

### GET INVOLVED

Email

Zip

### WHAT'S NEW

**April 4, 2012**

Governor Cuomo Announces NY Works Projects to Upgrade S...

**April 4, 2012**

Governor Cuomo Announces NY Works Projects to Upgrade S...

**April 4, 2012**

Governor Cuomo Announces NY Works Projects to Upgrade S...

 Connect to Gov. Cuomo on [facebook](#)

[www.governor.ny.gov](http://www.governor.ny.gov)





# OFFICE OF GENERAL SERVICES



Governor Andrew M. Cuomo

Commissioner RoAnn M. Destito

Authorized User Site

Vendor Site

HBITS Contracts – LINK Only

Vendor Information–NEXT PAGE

*Both would be secure/ password protected for the Authorized Users and the Vendors.*

## Hourly-Based Information Technology Services (HBITS)

This web portal is for Agencies, Customers and Vendors selected through the HBITS Contract # \_\_\_\_\_.

- A Task Order job cannot exceed 24 months (without exception).
- Please follow the process delineated for either Agency/Customer or Vendor. Each web portal is independent of one another and cannot be seen by the other party.

**\*\*This is the informational Screen that is the access to the secure portal. This is what the public can view.**

Questions should be addressed to the State’s HBITS Coordinator, \_\_\_\_\_, at \_\_\_\_\_.



**HBITS Vendor List**

- Authorized User Site
- Vendor Site
- HBITS Contracts - LINK
- Vendor Information-NEXT PAGE

PUBLIC SITE



- [1](#) Vendor
- [2](#) Vendor
- [3](#) Vendor
- [4](#) Vendor
- [5](#) Vendor
- [6](#) Vendor
- [7](#) Vendor
- [8](#) Vendor
- [9](#) Vendor
- [10](#) Vendor
- [11](#) Vendor
- [12](#) Vendor
- [13](#) Vendor
- [14](#) Vendor
- [15](#) Vendor
- [16](#) Vendor
- [17](#) Vendor
- [18](#) Vendor
- [19](#) Vendor
- [20](#) Vendor

The following list of vendors currently supports the HBITS contract.

Click on the link for additional information on the vendor.

All vendors will be re-assessed annually to adjust the waitlisted group of vendors as necessary.

**This area of the portal will contain the relevant information for the 25 vendors and would be viewable to the general public.**

**Waitlisted**

- [1](#) Vendor
- [2](#) Vendor
- [3](#) Vendor
- [4](#) Vendor
- [5](#) Vendor



Authorized User Information



## Hourly-Based Information Technology Services (HBITS)

Task Order Form

Preliminary Technical Evaluation Form

Interview Evaluation Form

Current Task Orders

### GENERAL INFORMATION FOR AUTHORIZED USERS

Questions should be addressed to the State's HBITS Coordinator, \_\_\_\_\_, at \_\_\_\_\_.



*Governor Andrew M. Cuomo*

# OFFICE OF GENERAL SERVICES



*Commissioner RoAnn M. Destito*

Agency Information

Task Order Form

Preliminary Technical  
Evaluation Form

Interview Evaluation Form

Current Task Orders



See **Form #1** in Exhibit 1

This is the web-based form authorized users will complete to initiate HBITS support.



# OFFICE OF GENERAL SERVICES



Governor Andrew M. Cuomo

Commissioner RoAnn M. Destito

Agency Information

Task Order Form

Preliminary Technical  
Evaluation Form

Interview Evaluation Form

Current Task Orders



See **Form #3A** in Exhibit 1

This is the web-based form authorized users will complete to complete HBITS candidate evaluations.



# OFFICE OF GENERAL SERVICES



*Governor Andrew M. Cuomo*

*Commissioner RoAnn M. Destito*

Agency Information

Task Order Form

Preliminary Technical  
Evaluation Form

Interview Evaluation Form

Current Task Orders



See **Form #3B** in Exhibit 1

This is the web-based form authorized users will complete to complete HBITS candidate evaluations.



# OFFICE OF GENERAL SERVICES



Governor Andrew M. Cuomo

Commissioner RoAnn M. Destito

Agency Information

Task Order Form

Preliminary Technical Evaluation Form

Interview Evaluation Form

Current Task Orders



## New York State Hourly IT Services Support Current Task Orders

Task Order #	Agency	Project Name	Job Title	# of Staff	Start Date	Est. Hours	Est. Completion Date	Est. End Date	Location
12-13-001	OCFS	Connections	JAVA General Maintenance	4	8/1/2012	200	4 weeks	9/1/2012	Albany
12-13-002	DEC	N/A	JAVA Improvements	3	8/15/2012	300	6 weeks	10/1/2012	Albany



*Governor Andrew M. Cuomo*

# OFFICE OF GENERAL SERVICES



*Commissioner RoAnn M. Destito*

Vendor Information



Task Order Bulletin Board  
(Current Bids)

Task Order Bid Form

## GENERAL VENDOR INFORMATION

Questions should be addressed to the State's HBITS Coordinator, \_\_\_\_\_, at \_\_\_\_\_.



Governor Andrew M. Cuomo

## OFFICE OF GENERAL SERVICES



Commissioner RoAnn M. Destito

Vendor Information

Task Order Bulletin Board

Candidate Response Form

## New York State HBITS Task Order Bulletin Board

Vendors interested in bidding on the posted Task Orders below must complete a TO Bid Form, which is available by clicking on the TO # below or completing the form, by clicking on the icon to the left. Task orders are posted for a period of 5-10 business days. At that time the TO will be removed from the bulletin board and bids will no longer be accepted for the posted TO. TO Closing Date is also posted on the bulletin board in the last column of each posted TO.

Task Order #	Agency	Job Title	# of Staff	Start Date	Est. Hours	Est. Completion Date	Est. End Date	Location	TO Closing Date
12-13-001	OCFS	JAVA General Maintenance	4	8/1/2012	200	9/1/2012	9/1/2012	Albany	5 days from Posting
12-13-002	DEC	JAVA Improvements	3	8/15/2012	300	10/1/2012	10/1/2012	Albany	5 days from Posting

  
Hyperlink



*Governor Andrew M. Cuomo*

# OFFICE OF GENERAL SERVICES



*Commissioner RoAnn M. Destito*

Vendor Information

Task Order Bulletin Board  
(Current Bids)

Candidate Response Form



See **Form #2** in Exhibit 1

This is the web-based form vendors will use to propose potential candidates for the posted Task Order jobs.