Novi Survey Application Programming Interface

Startup Guide

1. Introduction

Novi Survey includes both a full object API and a web service API. The object API is fined grained and allows for full control and management of the surveys, responses, invitations, etc. The web services API is coarse grained and provide common high level facilities such as support for single sign on or sending of email invitations.

2. Object API

The object API is available to customers with the source code option. The class description for the object API is included in `NoviSurvey Core Class Library.chm`. File `NoviSurvey.Core.dll` includes the object API. Additionally, files `NHibernate.dll`, `log4net.dll`, and `Iesi.Collections.dll` are needed to develop against the object API. All DLLs are available in the `bin` directory for the web application.

2.1 Establishing connection to the database

Use of the object API requires configuration data such that a database connection to the Novi Survey database can be acquired.

2.1.1 Form application or class library

The configuration is set by inserting the following statements in file `App.config`. In the configuration data, property `connection.connection_string` should be modified to reflect the actual runtime environment:

```
  <hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
    <session-factory>
      <property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>

      <property name="connection.driver_class">NHibernate.Driver.SqlClientDriver</property>
      <property name="dialect">NHibernate.Dialect.MsSql2000Dialect</property>
      <property name="connection.connection_string">Data Source=;Database=NoviSurvey;User
Id=NoviSurvey;Password=NoviSurvey;</property>

      <property name="cache.provider_class">NHibernate.Cache.HashtableCacheProvider,
NHibernate</property>
      <property name="cache.use_second_level_cache">false</property>
      <property name="cache.use_query_cache">false</property>

      <mapping assembly="NoviSurvey.Core" />
    </session-factory>
  </hibernate-configuration>
```

2.1.2 Web application

Create a file named `NHibernate.config` in the root directory for the web application. Set the file content to the following:

```
<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
  <session-factory>
    <property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>

    <property name="cache.provider_class">NHibernate.Cache.HashtableCacheProvider,
NHibernate</property>
    <property name="cache.default_expiration">7200</property>
    <property name="cache.use_second_level_cache">false</property>
    <property name="cache.use_query_cache">false</property>

    <mapping assembly="NoviSurvey.Core"/>
  </session-factory>
</hibernate-configuration>
```

In the web application, add the following to initialize the connection to the database:

```
          NHibernateSessionMgr.Instance.Driver = "NHibernate.Driver.SqlClientDriver";
          NHibernateSessionMgr.Instance.Dialect = "NHibernate.Dialect.MsSql2000Dialect";
          NHibernateSessionMgr.Instance.ConnectionString = @"Data Source=;Database=NoviSurvey;User
Id=NoviSurvey;Password=NoviSurvey;";
          NHibernateSessionMgr.Instance.InitSessionFactory();
```

Note that the value of `NHibernateSessionMgr.Instance.ConnectionString` for should be adjusted to reflect the runtime environment.

2.2 Sample object API interaction

The script below illustrates how to work with the object API. It creates a survey with one open ended question and one multiple choice question and saves the survey in the database.

```
      NHibernateSessionMgr.Instance.BeginTransaction();

      var folder = new FolderDao().FindById("1");
      var template = new TemplateDao().FindDefault();
      var culture = new CultureLocaleDao().FindByName("en-US");

      // create survey
      var newSurvey = new Survey {
        Folder = folder,
        DeploymentId = 1000,
        UserDeploymentId = "1000",
        Template = template,
        Status = Survey.SurveyStatus.Open
      };

      var surveyLoc = new SurveyLoc { Culture = culture, Name = "new survey", IsDefault = true };
      newSurvey.AddLocalized(surveyLoc);

      // add new page to survey
      var page = new SurveyPage();
      newSurvey.AddPage(page, 0);

      var pageLoc = new SurveyPageLoc { Culture = culture, Name = "page 1" };
      page.AddLocalized(pageLoc);
```

```
      // add open ended question to page
      var openEnded = new OpenEnded {
        ValueType = Answer.ValueType.Text
      };
      page.AddElement(openEnded);

      var openEndedLoc = new ElementLoc { Culture = culture, Name = "open ended question" };
      openEnded.AddLocalized(openEndedLoc);

      for (var i = 0; i < 5; i++) {
        var answer = new Answer { Question = openEnded };
        openEnded.Add(answer);
        var answerLoc = new AnswerLoc { Culture = culture, Name = "field" + i };
        answer.AddLocalized(answerLoc);
      }

      // add multiple choice question to page
      var multipleChoice = new MultipleChoice {
        FreeTextAnswerWidget = Cv.TextWidget.SingleLine,
        FreeTextEntryRequired = false
      };
      page.AddElement(multipleChoice);

      var multipleChoiceLoc = new ElementLoc { Culture = culture, Name = "test multiple choice
question" };
      multipleChoice.AddLocalized(multipleChoiceLoc);

      for (var i = 0; i < 5; i++) {
        var answer = new Answer { Element = multipleChoice };
        multipleChoice.Add(answer);
        var answerLoc = new AnswerLoc { Culture = culture, Name = "option" + i };
        answer.AddLocalized(answerLoc);
      }

      var freeTextAnswer = new Answer();
      multipleChoice.FreeTextAnswer = freeTextAnswer;
      var freeTextAnswerAnwerLoc = new AnswerLoc { Culture = culture, Name = "free text answer" };
      freeTextAnswer.AddLocalized(freeTextAnswerAnwerLoc);

      var dbSession = NHibernateSessionMgr.Instance.GetSession();
      dbSession.SaveOrUpdate(newSurvey);
      NHibernateSessionMgr.Instance.CommitTransaction();
```

3. Web services API

The web services API is available at endpoints:

`~/ws/AdminWebService.asmx`

`~/ws/SurveyWebService.asmx`

The services provided are described below. All samples calls are shown using C# and assume that the namespace for the web services stubs is `NoviSurvey`. In all calls "id" parameters (e.g., userId, orgId) refer to the value for the primary key in a database table row (e.g., userId should match NOVIUSER.USERID).

3.1 Admin Web Service

3.1.1 Establishment of a session

Calls to AdminWebService are stateful and require that a valid session be established. If a session is not established, the calls will fail. A session can be established as follows:

```
  public static NoviSurvey.AdminWebService GetNoviSurveyAdminService(string endpointUrl, string
adminPassword) {
    var url = endpointUrl + "/ws/AdminWebService.asmx";
    var svc = new NoviSurvey.AdminWebService {
      Url = url,
      CookieContainer = new CookieContainer()
    };
    var authSuccess = false;
    try {
      authSuccess = svc.Authenticate("admin", adminPassword);
    } catch (InvalidOperationException) {
      // no-op
    }
    // necessary to handle redirection in auto detect cookie mode
    if (!authSuccess) {
      authSuccess = svc.Authenticate("admin", adminPassword);
    }
    if (!authSuccess) {
      throw new Exception("Unable to authenticate for web service: " + url);
    }

    return svc;
  }
```

3.1.2 Authenticate

Purpose	Authenticates the caller. Authentication is required prior to making any other call to the AdminWebService. Authentication to an account with system administrator privileges is required to use the other administrative services.
Signature	`public bool Authenticate(string userName, string password)`
Sample call	See section 3.1.1

3.1.3 Create organization

Purpose	Creates a new organization.
Signature	`public string CreateOrganization(string organizationName)`

Sample call	`var svc = GetNoviSurveyAdminService(url, adminPassword);`
	
	`var orgId = svc.CreateOrganization(orgName);`

3.1.4 Create user

Purpose	Creates a user in an organization. The organization must pre-exist. See also 3.1.3.

Signature	`public string CreateUser(string organizationId, string firstName, string lastName, string userName, string email, string passwordHash)`
	
	`organizationId: maps to ORGANIZATION.ORGANIZATIONID`

Sample call	`var svc = GetNoviSurveyAdminService(url, adminPassword);`
	`var passwordHash = Convert.ToBase64String(new SHA1CryptoServiceProvider().ComputeHash(new UTF8Encoding().GetBytes(newUserPassword)));`
	`var userId = svc.CreateUser(orgId, firstName, lastName, userName, email, passwordHash);`

3.1.5 Update user

Purpose	Update a preexisting user. See also 3.1.4.
Signature	`public void UpdateUser(string userId, string firstName, string lastName, string email, string passwordHash)`
Sample call	`var svc = GetNoviSurveyAdminService(url, adminPassword);` `svc.UpdateUser(userId, newFirstName, newLastName, newEmail, newPasswordHash);`

3.1.6 Set user roles

Purpose	Set the security roles for a preexisting user. See also 3.1.4.

Signature	`public void SetUserRoles(string userId, string organizationId, string[] roleNames)`
	
	`roleNames: some subset of:`
	`{ "OrgAdminRole", "SurveyCreatorRole", "BulkDataEntryRole", "DeleteResponsesRole", "WebServiceCallerRole", "RestrictToPortalRole" }`
	
	`userId: maps to NOVIUSER.USERID`
	
	`organizationId: maps to ORGANIZATION.ORGANIZATIONID`

Sample call	`var svc = GetNoviSurveyAdminService(url, adminPassword);` `svc.SetUserRoles(userId, orgId, new[] { "SurveyCreatorRole", "BulkDataEntryRole", "DeleteResponsesRole" });`

3.1.7 Get login validation token

Purpose	Generates a login token that can be used to implement a single sign-on mechanism. The login token must be used within 45 s after it generated. After that time, the token will no longer be valid.
Signature	`public string GetLoginValidationToken()`

Sample call	`var svc = GetNoviSurveyAdminService(url, adminPassword);`
	`var loginToken = svc.GetLoginValidationToken().Replace(" ", "+");`
	`var url = string.Format("{0}/Login.aspx?ui={1}&vt={2}&ReturnUrl={3}",`
	`surveyBaseUrl,`
	`Server.UrlEncode(userId),`
	`Server.UrlEncode(loginToken),`
	`Server.UrlEncode("~/s/SurveyList.aspx"));`
	`Response.Redirect(url);`

3.2 Survey Web Service

All calls to survey web service return one of the two standardized responses shown below:

```
/// <summary>
/// A response to a service call
/// </summary>
public class ServiceResponse {
  /// <summary>
  /// True is the call was successful, False otherwise
  /// </summary>
  public bool Success;
  /// <summary>
  /// An optional message providing details on the outcome of the call
  /// </summary>
  public string Message;
}

/// <summary>
/// A response to a service call that returns data in the form of a list
/// </summary>
public class ServiceResponse<T> {
  /// <summary>
  /// True is the call was successful, False otherwise
  /// </summary>
```

```
    public bool Success;
    /// <summary>
    /// An optional message providing details on the outcome of the call
    /// </summary>
    public string Message;
    /// <summary>
    /// The data returned by the call. In general and unless specified otherwise for a particular data
call, if <see cref="Success"/> is false, then Data is null.
    /// </summary>
    public List<T> Data;
  }
```

3.2.1 Send survey invitation

Purpose	Allows to create and send an email invitation for a survey. The service is subject to the following constraints: 1/ The session should be authenticated though a call of AdminWebService.Authenticate prior to calling this service. 2/ Session management through cookie should be enabled. 3/ The authenticated user must be active and have role Role.WebServiceCallerRole to call this service. 4/ The organization containing the survey must be active. 5/ The invitation group must be active Additional constraints are listed for each of the parameters. If the invitation is created but cannot be sent due to a failure of the email connector, the invitation will be sent at a later time in the same manner as if it was created by through the user interface. The person invited will be added to one of the person lists associated with the invitation group. The same person list will be the same for a given invitation group.

Signature	`public ServiceResponse SendSurveyInvitation(`
	`  string surveyDeploymentId, string inviteeFirstName,`
	`  string inviteeLastName, string inviteeEmail,`
	`  string invitationGroupName,`
	`  string[] surveyParameterKeys,`
	`  string[] surveyParametersValues)`
	
	`/// <param name="surveyDeploymentId">The deployment Id for the survey. The deployment Id corresponds to the value of parameter 's' for the survey deployment URL shown in the deployment tab for the survey set up screen. The survey must be opened for responses.</param>`
	
	`<param name="inviteeFirstName">The first name for the invitee. The first name cannot be blank.</param>`
	
	`<param name="inviteeLastName">The last name for the invitee. The last name cannot be blank.</param>`
	
	`<param name="inviteeEmail">The email for the invitee. The email cannot be blank and must be a valid email address.</param>`
	
	`<param name="invitationGroupName">The name for the list of the invitation group where the invitee should be added. The invitation group must exist (see invitation tool in the survey list screen)</param>`
	
	`<param name="surveyParameterKeys">The keys for the parameters for the survey that will have`

	values for the invitation. The parameter keys must match the values defined in for response parameters the deployment tab in the survey set up screen.</param> <param name="surveyParametersValues">The values to associate the survey parameter keys</param> /// <returns>Standardized service response</returns>

Sample call	const string baseUrl = "http://survey.foo.com";
	const string baseServicesUrl = baseUrl + "/ws";
	const string userName = "[userName]";
	const string password = "[password]";
	
	var cookieJar = new CookieContainer();
	var wr = (HttpWebRequest)WebRequest.Create(baseUrl);
	wr.CookieContainer = cookieJar;
	wr.GetResponse().Close();
	
	var authWs = new NoviSurvey.AdminWebService { Url = baseServicesUrl + "/AdminWebService.asmx", CookieContainer = cookieJar};
	if (!authWs.Authenticate(userName, password)) {
	  throw new InvalidCredentialException("authentication failed with the credentials as supplied");
	}
	
	try {
	  var surveyWs = new NoviSurvey.SurveyWebService {Url = baseServicesUrl + "/SurveyWebService.asmx", CookieContainer = cookieJar};
	  var response = surveyWs.SendInvitation("[surveyDeploymentId]", "[inviteeFirsName]", "[inviteeLastName]", "[inviteeEmail]", "[invitationGroupName]", new string[] { }, new string[] { });
	  if (!response.Success) {
	  // error handling code for invitation creation failure
	  }
	} catch (Exception ex) {
	  // error handling for a failure of the call (e.g., network failure, service unavailable, ...)
	}

3.2.2 Get survey respondents

Purpose	Provides a list of respondents for a survey within some date range. The service is subject to the following constraints: 1/ The session should be authenticated though a call to AdminWebService.Authenticate prior to calling this service. 2/ Session management through cookie should be enabled. 3/ The authenticated user must be active and have role Role.WebServiceCallerRole to call this service. 4/ The organization containing the survey must be active. Additional constraints are listed for each of the parameters.

Signature	public ServiceResponse<NoviPerson> GetSurveyRespondents(
	  string surveyDeploymentId,
	  DateTime startDate,
	  DateTime endDate
	  )
	
	<param name="surveyDeploymentId">The deployment Id for the survey. The deployment Id corresponds to the value of parameter 's' for the survey deployment URL shown in the deployment tab for the survey set up screen. The survey must be opened for responses.</param>

	<param name="startDate">The earliest date when the respondents started the survey. The earliest date is from the beginning of the day. The earliest date must be at lease Jan-1-2000.</param> <param name="endDate">The latest date when the respondents started the survey. The latest date is to the end of the day.</param> <returns>Standardized service response. The data from the service is in the Data field as a list of NoviPerson</returns>
Sample call	`var respondents = NoviSurvey.GetSurveyRespondents(surveyId, new DateTime(2010, 9, 1), new DateTime(2010, 9, 30));`

3.2.3 Get survey responses

Purpose	Provides response data for a survey. The data is provided in the same way as in the export from the browse and report lists screens. The service is subject to the following constraints: 1/ The session should be authenticated though a call to AdminWebService.Authenticate prior to calling this service. 2/ Session management through cookie should be enabled. 3/ The authenticated user must be active and have role Role.WebServiceCallerRole to call this service. 4/ The organization containing the survey must be active. 5/ The service can be called at most 100 times per day. After this limit each reached, the calls will be returned with a failure message. 6/ The maximum size allowed for the returned data is 1 MB. If parameters are specified that result in a export larger than this size, the call will be returned with a failure message. Additional constraints are listed for each of the parameters.

Signature	`public ServiceResponse<string> GetSurveyResponses(`
	`  DateTime fromDate,`
	`  DateTime toDate,`
	`  string surveyDeploymentId,`
	`  ExportService.Separator separator,`
	`  string headerSeparator,`
	`  bool stripHeaderWhiteSpace,`
	`  bool condenseValues,`
	`  bool encodedHeaderFormat,`
	`  string tagForOptionsSelected,`
	`  string tagForOptionsNotSelected,`
	`  string tagForNas,`
	`  bool includePersonData,`
	`  bool includeParameterData,`
	`  bool includeScoreData,`
	`  bool includeResponseData,`
	`  bool includePartialPageData`
	`  )`
	
	`/// <param name="fromDate">The minimum value of the date last edited field for the responses provided.</param>`
	`/// <param name="toDate">The minimum value of the date last edited field for the responses provided. The toDate must be greater or equal to the fromDate.</param>`
	`/// <param name="surveyDeploymentId">The deployment Id for the survey, as found in the survey setup screen, deployment tab.</param>`
	`/// <param name="separator">The separator for values in the table.</param>`
	`/// <param name="headerSeparator">The separator to use when column headers are comprised of several sub elements (e.g., question name followed by matrix column name). Recommended value is "|"</param>`
	`/// <param name="stripHeaderWhiteSpace">Determines if white space is removed from the table column headers</param>`

	/// <param name="condenseValues">Determines if values in the table should be condensed. Requesting condensed value will result in fewer columns in the export. /// Some values for responses will be the aggregate of the choices made or the values entered by the participants.</param> /// <param name="encodedHeaderFormat">Determines if the column headers for the export are built of the question and question sub-elements text or are generated as codes (e.g., P1Q1)</param> /// <param name="tagForOptionsSelected">Determines the tag to use to indicate selection of an option (e.g., in a multiple choice question). This parameter applies only when values are not condensed.</param> /// <param name="tagForOptionsNotSelected">Determines the tag to use to indicate that an option was not selected (e.g., in a multiple choice question). This parameter applies only when values are not condensed.</param> /// <param name="tagForNas">Value to use to backfill cells with no values in the table.</param> /// <param name="includePersonData">Determines if participant information should be included in the table.</param> /// <param name="includeParameterData">Determines if response parameter information should be included in the table.</param> /// <param name="includeScoreData">Determines if scoring information should be included in the table.</param> /// <param name="includeResponseData">Determines if response meta data (e.g., response start and end time) should be included in the table.</param> /// <param name="includePartialPageData">Determines if response data from partial saves should be included in the export.</param> /// <returns>Standardized service response. The data from the service, if any, is in the first entry of the Data field as a string.</returns>

Sample call	var surveySvc = new NoviSurvey.SurveyWebService {CookieContainer = cookieContainer, Url = WebUtils.GetSystemUrl(null, "~/ws/SurveyWebService.asmx")};
	var data = surveySvc.GetSurveyResponses(
	new DateTime(2010, 7, 11),
	new DateTime(2010, 7, 17),
	"zzza",
	Separator.Comma,
	"|",
	true,
	true,
	true,
	"1",
	"0",
	"NA",
	true,
	true,
	true,
	true,
	true);
	
	string tabularData;
	if (!response.Success) {
	throw new Exception("export was not successful: " + response.Message);
	} else {
	tabularData = response.Data.First();
	}

4. Direct SQL Access

Some commonly used queries directly against the schema for Novi Survey are provided below.

4.1. Organization roles

Provides a list of all users, organization users, and their roles.

```
select nu.USERNAME, np.FIRSTNAME, np.LASTNAME, np.EMAILADDRESS, org.ORGANIZATIONNAME,
nu.SYSADMINFLAG, ou.ACTIVEFLAG, urm.ROLENAME
from ORGANIZATION org
inner join ORGANIZATIONUSER ou on ou.ORGANIZATIONID = org.ORGANIZATIONID
inner join NOVIUSER nu on ou.USERID = nu.USERID
inner join NOVIPERSON np on nu.NOVIPERSONID = np.NOVIPERSONID
left outer join USERROLEMAP urm on urm.ORGANIZATIONUSERID = org.ORGANIZATIONUSERID
```

4.2 Survey privileges

Provides a list of all surveys and the privileges that organization users have to these surveys.

```
-- survey privilege query
select nu.USERNAME, np.FIRSTNAME, np.LASTNAME, np.EMAILADDRESS, org.ORGANIZATIONNAME,
sl.SURVEYNAME,
case sp.PRIVILEGE
when 0 then 'Owner'
when 1 then 'Designer'
when 2 then 'Analyst'
when 3 then 'Invitation sender'
else 'unknown'
end PRIVILEGE
from ORGANIZATION org
inner join ORGANIZATIONUSER ou on ou.ORGANIZATIONID = org.ORGANIZATIONID
inner join NOVIUSER nu on ou.USERID = nu.USERID
inner join NOVIPERSON np on nu.NOVIPERSONID = np.NOVIPERSONID
inner join FOLDER f on f.ORGANIZATIONID = org.ORGANIZATIONID
inner join SURVEY svy on svy.FOLDERID = f.FOLDERID
inner join SURVEYLOC sl on sl.SURVEYID = svy.SURVEYID and sl.DEFAULTCULTLOCFLAG = 'T'
inner join SURVEYPRIVILEGE sp on sp.SURVEYID = svy.SURVEYID and sp.ORGANIZATIONUSERID =
ou.ORGANIZATIONUSERID
```

4.3 Survey response data

Survey response data is available from view QUESTIONRESPONSE_V. The view is not available by default in the schema and must be created.