Medidata.RWS.NET.Standard Documentation

Release 1.0

Matthew Koch, MSKCC

Aug 16, 2018

Contents

1 Introduction **3**
- 1.1 Assumptions . . . 3
- 1.2 Acknowledgments . . . 3

2 Getting Started **5**
- 2.1 Installation . . . 5
- 2.2 Basic Example . . . 5
- 2.3 Miscellaneous Configuration . . . 6

3 Topics **9**
- 3.1 Authentication . . . 9
- 3.2 Errors . . . 9

4 Basic Requests **11**
- 4.1 VersionRequest() . . . 11
- 4.2 TwoHundredRequest() . . . 11

5 Core Resources **13**
- 5.1 Clinical Data . . . 13
- 5.2 Clinical View Datasets . . . 14

6 Using Builders **19**
- 6.1 Basic Example - Register a Subject . . . 19

7 Indices and tables **21**

Medidata.RWS.NET.Standard is a comprehensive, fluent .NET Standard 2.0 API library for Medidata RAVE Web Services (RWS). It handles a large portion of the boilerplate C# code you'd normally have to write in order to communicate with RWS, allowing you to get up and running faster.

CHAPTER 1

Introduction

1.1 Assumptions

This documentation assumes that the reader has a familiarity with the Medidata RAVE platform, as well as a basic understanding of RESTful Web Service concepts. While the Medidata.RWS.NET.Standard library attempts to abstract away some of the typical HTTP communication mechanisms required when making typical web service requests, the underlying HTTP codes and raw request / response data is still available to developers.

1.2 Acknowledgments

The creation of this library wouldn't have been possible without inspiration and examples from some fine folks at Medidata Solutions, specifically Ian Sparks and Geoff Low. In many ways, the library has taken some inspiration from the Python flavored "rwslib", available on Github: https://github.com/mdsol/rwslib.

CHAPTER 2

Getting Started

2.1 Installation

Install Medidata.RWS.NET.Standard via NuGet: `nuget install Medidata.RWS.NET.Standard`

or, via the Package Manager Console: `Install-Package Medidata.RWS.NET.Standard`

2.2 Basic Example

At the most basic level, communicating with Medidata RAVE using Medidata.RWS.NET.Standard involves:

1. Creating a connection.

```
using Medidata.RWS.NET.Standard.Core.Requests;

var connection = new RwsConnection("innovate", "RAVE username", "RAVE password");
```

The above code will create a RwsConnection object to the "innovate" RAVE instance (subdomain) - https://innovate.mdsol.com. You'd substitute your RAVE instance subdomain here. The RAVE username and RAVE password parameters should reference a dedicated RAVE account you intend to use for web service activities.

2. Creating a request.

```
using Medidata.RWS.NET.Standard.Core.Requests.Datasets;

var datasetRequest = new SubjectDatasetRequest("MediFlex", "PROD", subject_key:
↪"SUBJECT001", formOid: "HEM");
```

3. Sending the request.

```
var response = await connection.SendRequestAsync(datasetRequest) as RwsResponse;
```

4. Dealing with the response/exception.

```
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());
```

5. Putting this all together, we have the following.

```
    //Create a connection
    var connection = new RwsConnection("innovate", "RAVE username", "RAVE password");

    //Create a request
    var datasetRequest = new SubjectDatasetRequest("MediFlex", "PROD", subject_key:
↪"SUBJECT001", formOid: "HEM");

    //Send the request / get a response
var response = await connection.SendRequestAsync(datasetRequest) as RwsResponse;

    //Write the response XML string to the console
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());
```

The above steps outline how to retrieve data for **SUBJECT001**'s **HEM** form in the **MediFlex** study (specifically, the **PROD** environment).

2.3 Miscellaneous Configuration

When working with Medidata.RWS.NET.Standard, you may wish to configure the connection to RWS beyond the default settings. Options for doing so are described below.

2.3.1 Virtual Directory

When you make requests to RAVE Web services, the URL you communicate with follows a specific pattern. The domain name will always be "mdsol.com", and the protocol will always be "https://". The sub-domain & virtual directory are configurable. The default virtual directory is "RaveWebServices". This means that when you create a new connection to innovate using the default virtual directory, you'll get the following:

```
using Medidata.RWS.NET.Standard.Core.Requests;

//Create a connection
var connection = new RwsConnection("innovate", "RAVE username", "RAVE password");
connection.base_url; // 'https://innovate.mdsol.com/RaveWebServices'
```

If instead you'd like to change the virtual directory to a custom one, you can pass an additional parameter through the RwsConnection constructor:

```
using Medidata.RWS.NET.Standard.Core.Requests;

//Create a connection
var connection = new RwsConnection("innovate", "RAVE username", "RAVE password",
↪"CustomVirtualDirectory");
connection.base_url; // 'https://innovate.mdsol.com/CustomVirtualDirectory'
```

Note that this is provided as a convenience, and the default virtual directory will be preferable in most scenarios.

2.3.2 Timeouts

If supplied, the timeout of the request in milliseconds. If the request takes longer than the timeout value, an exception will be thrown.

```
using Medidata.RWS.NET.Standard.Core.Requests;

//Create a connection
var connection = new RwsConnection("innovate", "RAVE username", timeout: 1000)
```

CHAPTER 3

Topics

3.1 Authentication

In order to use Medidata RAVE web services, you must authenticate your web service requests. You can do this by supplying both a RAVE username and RAVE password parameter when you establish a `RwsConnection` object. This username/password should reference a dedicated RAVE account you intend to use for web service activities.

For example:

```
using Medidata.RWS.NET.Standard.Core.Requests;

var connection = new RwsConnection("innovate", "rwsUser1", "password1");
```

The above code will create a `RwsConnection` object and point it to the innovate RAVE instance (subdomain) - `https://innovate.mdsol.com`. The username and password you provide are concatenated, base64-encoded, and passed in the Authorization HTTP header each time you make a request using the connection object, as follows:

```
Authorization:  Basic cndzVXNlcjE6cGFzc3dvcmQx
```

Do not share your username / password in publicly accessible areas such GitHub, client-side code, and so forth. Authentication is only required when establishing a new `RwsConnection` object. The Medidata.RWS.NET.Standard library will then automatically send the appropriate Authorization header for each request made with this connection object.

API requests without authentication will fail.

3.2 Errors

Medidata RAVE web services uses HTTP response codes to indicate success or failure of an API request. Generally speaking, HTTP codes in the ranges below mean the following:

- 2xx - success
- 4xx - an error that failed given the information provided

- 5xx - an error with Medidata's servers

Not all errors map cleanly onto HTTP response codes, however. Medidata usually attempts to return a "RWS Reason Code" in addition to the conventional HTTP code to explain in more detail about what went wrong. To see a full listing of these codes, refer to: Rave Web Services Error Responses - Complete List

3.2.1 Handling errors

The Medidata.RWS.NET.Standard library can raise an exception for a variety reasons, such as invalid parameters, authentication errors, and network unavailability. We recommend writing code that gracefully handles all possible API exceptions.

CHAPTER 4

Basic Requests

Medidata.RWS.NET.Standard provides some basic diagnostic / health check API requests out of the box.

4.1 VersionRequest()

Returns the RWS version number. Specifically, this is the textual response returned when calling `https://{ subdomain }.mdsol.com/RaveWebServices/version`.

```
//Create a connection
var connection = new RwsConnection("innovate"); // no authentication required

//Send the request / get a response
var response = await connection.SendRequestAsync(new VersionRequest()) as␣
↪RwsTextResponse;

//Write the response text to the console
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());

//1.16.0
```

4.2 TwoHundredRequest()

Returns the html document (along with a 200 HTTP response code) that contains information about the MAuth configuration of Rave Web Services with the given configuration. Specifically, this is the html response returned when calling `https://{ subdomain }.mdsol.com/RaveWebServices/twohundred`.

```
//Create a connection
var connection = new RwsConnection("innovate"); // no authentication required

//Send the request / get a response
```

(continues on next page)

(continued from previous page)

```
var response = await connection.SendRequestAsync(new TwoHundredRequest()) as␣
↪RwsTextResponse;

//Write the response text to the console
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());

//<!DOCTYPE html>\r\n<html>\r\n<head><script..........
```

CHAPTER 5

Core Resources

5.1 Clinical Data

Often times you'll want to work with the clinical data of your studies. The Medidata.RWS.NET.Standard library provides several requests for extracting clinical data (via RAVE "Clinical Views") or POSTing clinical data to the RAVE platform.

5.1.1 ClinicalStudiesRequest

Returns a list of EDC studies (as a `RWSStudies` object). Excludes studies that you (the authenticated user) are not associated with.

Example:

```
    using Medidata.RWS.NET.Standard.Core.Requests;

    //Create a connection
    var connection = new RwsConnection("innovate", "username", "password"); //␣
↪authentication required

//Send the request / get a response
var response = await connection.SendRequestAsync(new ClinicalStudiesRequest()) as␣
↪RwsStudies;

    //Write the study list to the console
    foreach (var s in response)
    {
        Console.Write(s.OID + "\r\n");
    }
    //Mediflex(Prod)
    //Mediflex(Dev)
    //PlaceboTest(Prod)
    //...
```

5.1.2 StudySubjectsRequest

Returns a listing of all the subjects in a study (as a `RWSSubjects` object), optionally including those currently inactive or deleted. Clinical data for the subjects is not included in the response.

This is the equivalent of calling: `https://{subdomain}.mdsol.com/studies/{study-oid}/Subjects[?status=all&include={inactive|inactiveAndDeleted}]`

Parameters

Parameter	Description	Mandatory?
{study-oid}	The study name.	Yes
status={**true** | **false**}	If true, add subject level workflow status to the response (if present).	No
include= {**inactive** | **deleted** | **inactiveAndDeleted**}	Will include active, inactive and/or deleted subjects in the response.	No
subjectKeyType= {**SubjectName** | **SubjectUUID**}	Whether RWS should return the unique identifier (UUID) or the subject name.	No
links={**true** | **false**}	If true, includes "deep link"(s) (e.g. URLs) to the subject page in Rave in the response.	No

Example:

```
    using Medidata.RWS.NET.Standard.Core.Requests;

    //Create a connection
    var connection = new RwsConnection("innovate", "username", "password"); //␣
↪authentication required

//Send the request / get a response
var response = await connection.SendRequestAsync(new StudySubjectsRequest("Mediflex",
↪"Prod")) as RwsSubjects;

    //Write each subject key to the console
    foreach (var s in response)
    {
        Console.Write(s.SubjectKey + "\r\n");
    }
    // SUBJECT001
    // SUBJECT002
    // SUBJECT003
    // ...
```

5.2 Clinical View Datasets

In addition to the above requests, Medidata RAVE Web Services allows for the extraction of clinical data in the form of "Clinical Views" - that is, RAVE database views. There are 3 "Datasets" available that represent different subsets of clinical data for your studies:

5.2.1 StudyDatasetRequest

Clinical data in ODM format for the given study / environment. This data can be optionally filtered by a specific Form.

This is the equivalent of calling: `https://{subdomain}.mdsol.com/studies/{project}({environment})/datasets/{ regular|raw }?{options}`

or, to filter the data by form: `https://{subdomain}.mdsol.com/studies/{project}({environment})/datasets/{ regular|raw }/{ formoid }?{options}`

Example:

```
    //Create a connection
    var connection = new RwsConnection("innovate", "username", "password"); //
↪authentication required


//Send the request / get a response
var response = await connection.SendRequestAsync(new StudyDatasetRequest("Mediflex",
↪"Prod", dataset_type: "regular")) as RwsResponse;


    //Write the XML response to the console (see XML below)
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());
```

```
<?xml version="1.0" encoding="utf-8"?>
<ODM FileType="Snapshot" FileOID="92747321-c8b3-4a07-a874-0ecb53153f20"
↪CreationDateTime="2017-06-05T13:09:33.202-00:00" ODMVersion="1.3" xmlns:mdsol=
↪"http://www.mdsol.com/ns/odm/metadata" xmlns:xlink="http://www.w3.org/1999/xlink"
↪xmlns="http://www.cdisc.org/ns/odm/v1.3">
   <ClinicalData StudyOID="Mediflex(Prod)" MetaDataVersionOID="1">
        <SubjectData SubjectKey="1">
            <SiteRef LocationOID="1" />
            <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
                <FormData FormOID="CHEM" FormRepeatKey="1">
                    <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                        <ItemData ItemOID="CHEM.DATECOLL" Value="2015-04-25T14:09:00"
↪/>
                    </ItemGroupData>
                </FormData>
            </StudyEventData>
        </SubjectData>
    </ClinicalData>
    <ClinicalData StudyOID="Mediflex(Prod)" MetaDataVersionOID="1">
        <SubjectData SubjectKey="2">
            <SiteRef LocationOID="1" />
            <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
                <FormData FormOID="CHEM" FormRepeatKey="1">
                    <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                        <ItemData ItemOID="CHEM.DATECOLL" Value="2015-04-13T16:34:00"
↪/>
                    </ItemGroupData>
                </FormData>
            </StudyEventData>
        </SubjectData>
    </ClinicalData>
    <ClinicalData StudyOID="Mediflex(Prod)" MetaDataVersionOID="1">
        <SubjectData SubjectKey="3">
```

(continues on next page)

(continued from previous page)

```
            <SiteRef LocationOID="1" />
            <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
                <FormData FormOID="CHEM" FormRepeatKey="1">
                    <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                        <ItemData ItemOID="CHEM.DATECOLL" Value="2015-05-09T18:52:00"␣
↪/>
                    </ItemGroupData>
                </FormData>
            </StudyEventData>
        </SubjectData>
    </ClinicalData>
    ...
</ODM>
```

5.2.2 SubjectDatasetRequest

Clinical data in ODM format for the given study / environment for a single subject. Similar to `StudyDatasetRequest`, this data can be optionally filtered by a specific Form.

This is the equivalent of calling: `https://{subdomain}.mdsol.com/studies/{project}({environment})/subjects/{ subjectkey }/datasets/{ regular|raw }?{options}`

or, to filter the data by form: `https://{subdomain}.mdsol.com/studies/{project}({environment})/subjects/{ subjectkey }/datasets/{ regular|raw }/{ formoid }?{options}`

```
using Medidata.RWS.NET.Standard.Core.Requests
using Medidata.RWS.NET.Standard.Core.Requests.Datasets;

//Create a connection
var connection = new RwsConnection("innovate", "username", "password"); //␣
↪authentication required


//Send the request / get a response
var response = await connection.SendRequestAsync(new SubjectDatasetRequest("Mediflex",
↪ "Prod", subject_key: "1", dataset_type: "regular")) as RwsResponse;


//Write the XML response to the console (see XML below)
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());
```

```
<?xml version="1.0" encoding="utf-8"?>
<ODM FileType="Snapshot" FileOID="9035596c-f090-4030-860a-0ed27a4e3d03"␣
↪CreationDateTime="2017-06-05T13:28:39.325-00:00" ODMVersion="1.3" xmlns:mdsol=
↪"http://www.mdsol.com/ns/odm/metadata" xmlns:xlink="http://www.w3.org/1999/xlink"␣
↪xmlns="http://www.cdisc.org/ns/odm/v1.3">
<ClinicalData StudyOID="Mediflex(Prod)" MetaDataVersionOID="1">
    <SubjectData SubjectKey="1">
        <SiteRef LocationOID="1" />
        <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
            <FormData FormOID="CHEM" FormRepeatKey="1">
                <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                    <ItemData ItemOID="CHEM.DATECOLL" Value="2015-04-25T16:09:00" />
```

(continues on next page)

(continued from previous page)

```
                </ItemGroupData>
            </FormData>
        </StudyEventData>
    </SubjectData>
</ClinicalData>
<ClinicalData StudyOID="Mediflex(Prod)" MetaDataVersionOID="1">
    <SubjectData SubjectKey="1">
        <SiteRef LocationOID="1" />
        <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
            <FormData FormOID="ABX" FormRepeatKey="1">
                <ItemGroupData ItemGroupOID="ABX_LOG_LINE">
                    <ItemData ItemOID="ABX.ABXDATE" Value="2017-04-25" />
                    <ItemData ItemOID="ABX.MODALITY" Value="2" />
                </ItemGroupData>
            </FormData>
        </StudyEventData>
    </SubjectData>
</ClinicalData>
<ClinicalData StudyOID="Mediflex(Prod)" MetaDataVersionOID="1">
    <SubjectData SubjectKey="1">
        <SiteRef LocationOID="1" />
        <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
            <FormData FormOID="BONEMARROW" FormRepeatKey="1">
                <ItemGroupData ItemGroupOID="BONEMARROW_LOG_LINE">
                    <ItemData ItemOID="BONEMARROW.VISITDAT" Value="2015-04-24" />
                    <ItemData ItemOID="BONEMARROW.CHEMSAMPLE" Value="1" />
                    <ItemData ItemOID="BONEMARROW.BMPB_COLLECT" Value="1" />
                                                ...
                </ItemGroupData>
            </FormData>
        </StudyEventData>
    </SubjectData>
</ClinicalData>
    ...
</ODM>
```

5.2.3 VersionDatasetRequest

Clinical data in ODM format for the given study / environment for a single RAVE study version for all subjects. Similar to `StudyDatasetRequest`, this data can be optionally filtered by a specific Form.

This is the equivalent of calling: `https://{subdomain}.mdsol.com/studies/{project}({environment})/versions/{ version_id }/datasets/{ regular|raw }?{options}`

or, to filter the data by form: `https://{subdomain}.mdsol.com/studies/{project}({environment})/versions/{ version_id }/datasets/{ regular|raw }/{ formoid }?{options}`

```
using Medidata.RWS.NET.Standard.Core.Requests
using Medidata.RWS.NET.Standard.Core.Requests.Datasets;

//Create a connection
var connection = new RwsConnection("innovate", "username", "password"); //␣
↪authentication required
```

(continues on next page)

(continued from previous page)

```
//Send the request / get a response
var response = await connection.SendRequestAsync(new VersionDatasetRequest(project_
↪name: "Mediflex", environment_name: "Dev", version_oid: "999")) as RwsResponse;


//Write the XML response to the console (see XML below)
Console.Write(await response.ResponseObject.Content.ReadAsStringAsync());
```

*Note the **MetaDataVersionOID* value in the XML response.*

```
<?xml version="1.0" encoding="utf-8"?>
<ODM FileType="Snapshot" FileOID="9035596c-f090-4030-860a-0ed27a4e3d03"␣
↪CreationDateTime="2017-06-05T13:28:39.325-00:00" ODMVersion="1.3" xmlns:mdsol=
↪"http://www.mdsol.com/ns/odm/metadata" xmlns:xlink="http://www.w3.org/1999/xlink"␣
↪xmlns="http://www.cdisc.org/ns/odm/v1.3">
<ClinicalData StudyOID="Mediflex(Dev)" MetaDataVersionOID="999">
    <SubjectData SubjectKey="1">
        <SiteRef LocationOID="1" />
        <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
            <FormData FormOID="CHEM" FormRepeatKey="1">
                <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                    <ItemData ItemOID="CHEM.DATECOLL" Value="2015-04-25T16:09:00" />
                </ItemGroupData>
            </FormData>
        </StudyEventData>
    </SubjectData>
</ClinicalData>
<ClinicalData StudyOID="Mediflex(Dev)" MetaDataVersionOID="999">
   <SubjectData SubjectKey="2">
        <SiteRef LocationOID="1" />
        <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
            <FormData FormOID="CHEM" FormRepeatKey="1">
                <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                    <ItemData ItemOID="CHEM.DATECOLL" Value="2016-04-25T16:09:00" />
                </ItemGroupData>
            </FormData>
        </StudyEventData>
    </SubjectData>
</ClinicalData>
<ClinicalData StudyOID="Mediflex(Dev)" MetaDataVersionOID="999">
    <SubjectData SubjectKey="3">
        <SiteRef LocationOID="1" />
        <StudyEventData StudyEventOID="SCREENING" StudyEventRepeatKey="1">
            <FormData FormOID="CHEM" FormRepeatKey="1">
                <ItemGroupData ItemGroupOID="CHEM_LOG_LINE">
                    <ItemData ItemOID="CHEM.DATECOLL" Value="2017-04-25T16:09:00" />
                </ItemGroupData>
            </FormData>
        </StudyEventData>
    </SubjectData>
</ClinicalData>
    ...
</ODM>
```

CHAPTER 6

Using Builders

When communicating with Medidata RAVE Web Services, your data payloads will take the form of ODM XML - or Operational Data Model XML documents.

It's important to understand that RWS expects the XML data you send to conform to the ODM format - malformed or otherwise improperly formatted XML won't be processed. Since creating these data structures manually can be time consuming and tedious, Medidata.RWS.NET.Standard provides several "Builder" classes to help.

6.1 Basic Example - Register a Subject

By way of example, let's say you want to register a subject onto a RAVE study using RWS. In order to do this, you'll need:

1. An authenticated connection to RWS
2. An XML document that represents the POST request you intend to make, which will include:
   - A Study OID (study)
   - A LocationOID (site)
   - A SubjectKey (subject)
3. A way to deal with the response after the request is sent

The `ODMBuilder` class allows developers to build out the ODM XML documents required for transmission using a simple to use, fluent interface. Using the above example, let's create a new `ODMBuilder` instance to register a subject:

```
var odmObject = new ODMBuilder().WithClinicalData("MediFlex", cd =>
    cd.WithSubjectData("SUBJECT001", "SITE01", sd =>
        sd.WithTransactionType(TransactionType.Insert)));
```

After instantiating the `ODMBuilder` class, you'll notice that you have access to chain-able methods which allow you to construct the object appropriate for your use case. Since we are registering a subject, we supplied a Study OID (`Mediflex`), Subject Key (`SUBJECT001`), and Site (`SITE01`).

Each of the nested methods used (e.g. `WithClinicalData`, `WithSubjectData`, and `WithTransactionType`) map to the specific XML node we want to construct.

To see the XML string representation of what we've got so far, you can use the `AsXMLString()` method, which will convert the ODM object you constructed into an XML string.

For example:

```
string registrationXml = new ODMBuilder().WithClinicalData("MediFlex", cd =>
    cd.WithSubjectData("SUBJECT001", "SITE01", sd =>
        sd.WithTransactionType(TransactionType.Insert))).AsXMLString();
```

would produce:

```
<?xml version="1.0" encoding="utf-16"?>
<ODM xmlns:mdsol="http://www.mdsol.com/ns/odm/metadata" FileType="Transactional"␣
↪Granularity="All" FileOID="1d84fb20-1959-45bf-b9c4-cf2ad7a4273d" CreationDateTime=
↪"2017-09-14T15:01:50.8441121-04:00" AsOfDateTime="0001-01-01T00:00:00" ODMVersion=
↪"1.3" xmlns="http://www.cdisc.org/ns/odm/v1.3">
  <ClinicalData StudyOID="MediFlex" MetaDataVersionOID="1">
        <SubjectData SubjectKey="SUBJECT001" TransactionType="Insert">
          <SiteRef LocationOID="SITE01" />
        </SubjectData>
        <AuditRecords />
        <Signatures />
        <Annotations />
  </ClinicalData>
</ODM>
```

Using this ODM conformant XML, you could now POST it to RAVE by wrapping it in a `PostDataRequest` object:

```
//Create a connection
var connection = new RwsConnection("innovate", "username", "password");

//Send the request / get a response
var response = await connection.SendRequestAsync(new␣
↪PostDataRequest(registrationXml)) as RwsPostResponse;

Console.Write(response.SubjectNumberInStudy); // 12345

//If successful, SUBJECT001 should be registered in SITE01 for the Mediflex study.
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search