Ascio DNS Service
Reference Documentation

Revision 1.0

November 2010

Table of Contents

List of Figures

1 Introduction

This document provides reference material for use with the Ascio DNS Service (ADNSS). The Ascio DNS Service allows you to interface with the Ascio DNS. By using that interface, Ascio DNS Partner will be able to interact with the Ascio DNS Service. An Ascio DNS account is required in order to connect to the ADNSS. Once you have that, you will be able to develop your own solutions, e.g. web pages for you and your customers to manage the Ascio DNS portfolio.

Your staff members and customers may also login to ADNSS with their separate accounts and view and/or manage (based on their role types) your Ascio DNS portfolio.

This document includes example code for integrating ADNSS using the following technologies:

- .NET (with Visual Studio)

Note that ADNNS is not limited to these platforms and can be utilised through any other technology that supports SOAP.

2 Conceptual Overview

2.1 Roles and Users

Ascio DNS partner has master administrator rights. That means he/she can view and manage everything within his/her Ascio DNS portfolio. There can only be one level of users under Ascio DNS partner account. Every user must have a role assigned to him/her. Only one role can be assigned to a user. The roles are linked to one or more pre-defined user rights that allow doing certain kind of operations. E.g. Update any zone within partner portfolio, create users within partner portfolio etc. Every user has read access to his/her portfolio and can update his/her details e.g. email, password etc.

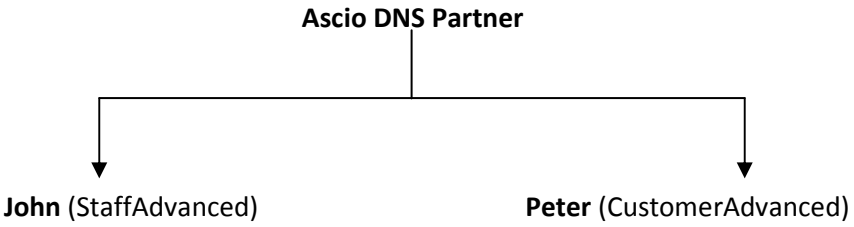

Figure 1: Single sub-level users with different roles

The rights associated to a role can be accessed via ADNSS, so that it is possible for the ADNSS clients to create a restricted/user-rights based menus in their applications.

2.1.1 Roles level and description

Role Level	Description	Example
StaffAdvanced	Typically for Ascio DNS Partner's staff members. User having this role can read, search and manage users and zones within the partner portfolio. However, cannot update/delete partner account details e.g. email, password etc. Also can only create another user with StaffBasic, CustomerAdvanced or CustomerBasic role.	John is a staff member of an Ascio DNS partner with StaffAdvanced role. He can view all zones and users within the partner portfolio. He can create, update, and delete users and zones within the partner portfolio. New user will be created under the partner's account, not as a sub-user of the John.
StaffBasic	Typically for Ascio DNS Partner's staff members. User having this role can read and search users and zones within the partner portfolio.	John is a staff member of an Ascio DNS partner with StaffBasic role. He can view all zones and users within the partner portfolio. No updates except John's own name, email

		and password are allowed.
CustomerAdvanced	Typically for the partner's customers User having this role can read, create, update, delete and search zones within own portfolio.	John is a customer of an Ascio DNS partner with CustomerAdvanced role. He can view, search and manage all his/her zones.
CustomerBasic	Typically for the partner's customers User having this role can read and search zones within own portfolio.	John is a customer of an Ascio DNS partner with CustomerBasic role. He can only view and search his/her zones.

2.2 Users and Zones

Zones can directly be linked to the Ascio DNS partner or one of its users. If a user own any zones and needs to be deleted, then zones must be deleted or moved to another user's account or Ascio DNS partner's account himself before deleting the user. Users with StaffBasic and StaffAdvanced roles can view and manage (applicable for StaffAdvanced role) other users' zones within the partner portfolio.

2.3 Headers

Ascio DNS Service requires at least two headers with correct values to be passed in each request in order to login successfully. In addition to that, one more header is required if partner's users (staff member or customers) need to login. Below is the description of the different headers.

Header Name	Description	
UserName	Ascio DNS partner or its customers username	Required in each request
Password	Ascio DNS partner or its customers password	Required in each request
Account	If partner's user (staff member or customer) needs to login then this header must be passed with the partner's username.	Required in each request if partner's users needs to login. Otherwise is optional.
TrackingReference	This is provided by the ADNSS client and is passed between request (in header) and response (in Response object). ADNSS stores the TrackingReference in its log files. So, in some cases partners may contact Ascio Partner Service to know what was done against the specific TrackingReference.	Optional

2.3.1 Set Headers - Example

.NET

```
public static void SetHeaders(string userName, string password, string account, string
trackingReference){


        const string nameSpace = "http://groupnbt.com/2010/10/30/Dns/DnsService";

OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("UserName",
nameSpace,
                                                    userName));

OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("Password",
nameSpace,
                                                    password));

OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("Account",
                                                    nameSpace, account));

OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("TrackingRefe
rence",
                                                    nameSpace, trackingReference));
    }
```

3 3 Getting Started

3.1 .NET / Visual Studio

Please note that the information in this section is targeted at the .NET Framework 4.0 and Visual Studio 2010.

This section will take you through the steps required to create a program that pass headers containing login details e.g. username and password to ADNSS. Passing the headers the program will call the GetUser method of the service.

3.1.1 Create a Project

In Visual Studio, go to *File -> New -> Project*. Select *Console Application, then rename the project to* MyDnsClient and click Ok. A new project MyDnsClient will be created.

3.1.2 Add a Reference to ADNSS

In the Solution Explorer as shown below, right-click the *References* item and select *Add Service Reference*.

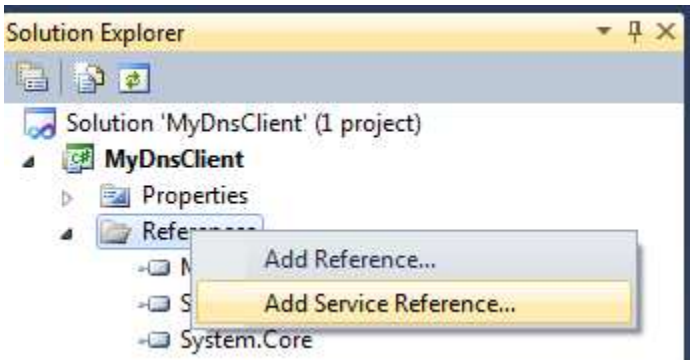

Figure 2: Solution Explorer

A dialogue is opened, see figure below. In the *Address* field, paste in the relevant URL, and click the *Go* button. DnsService should be found under the *Services* field. Rename the *Namespace* field value to DnsServiceTest as shown in the figure below and click OK. Visual Studio will now make ADNSS available to program against.

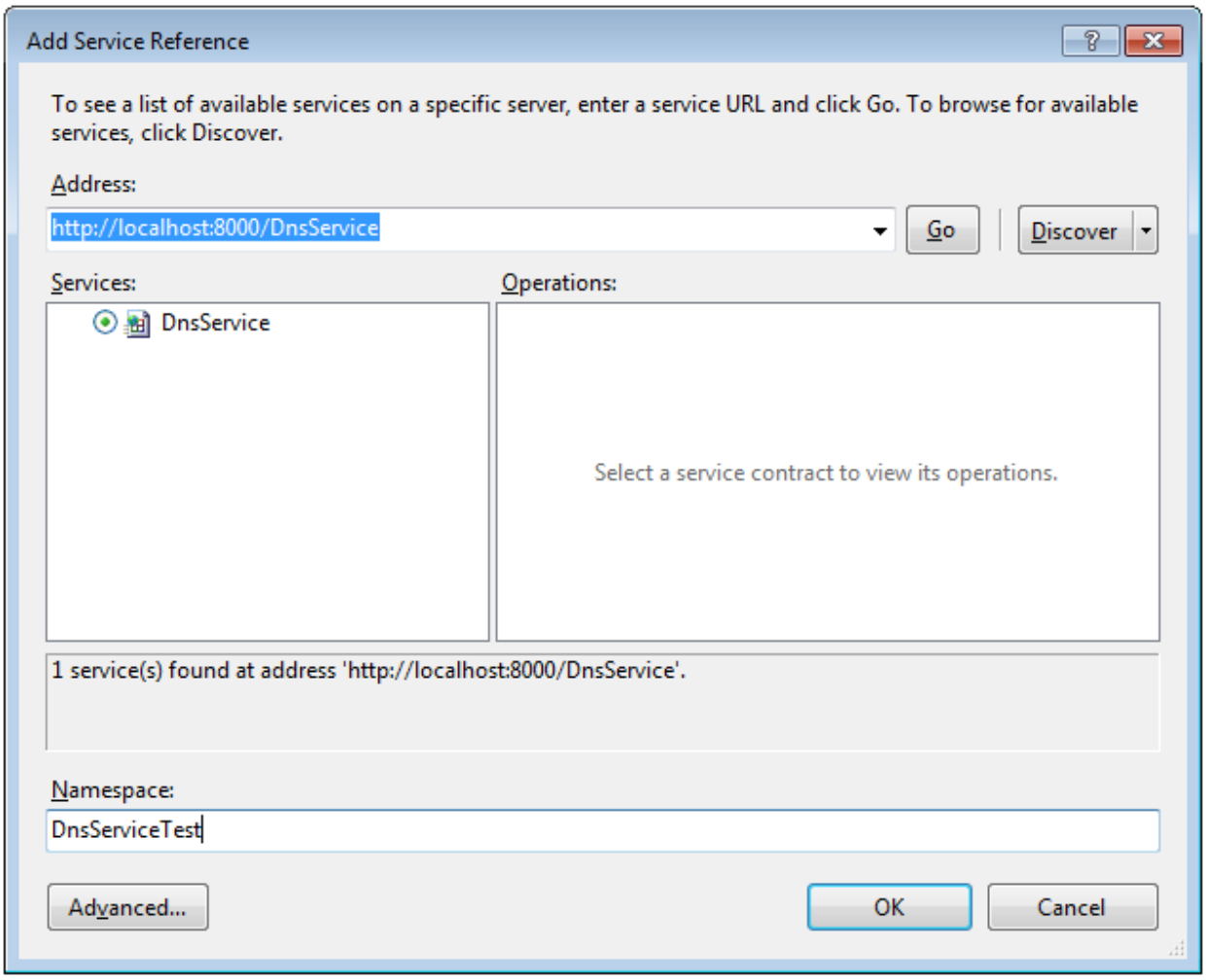

Figure 3: Add Service Reference

3.1.3 Insert a "Using" Directive

In order to make your code briefer and more readable, you should add a using directive:

Click on the Program.cs tab and insert the following line at the end of the using block at the top:

```
using MyDnsClient.DnsServiceTest;
```

3.1.4 Complete Example

Here is a complete code example. The user credentials are passed through the headers and GetUser method is called that returns the Response and User objects. There are different ways to pass the headers to a service in .NET 4 e.g. by implementing the IClientMessageInspector and IEndpointBehavior interfaces, and then add the custom Behavior to the target EndPoint of the service. In this example, an attempt has been made to add the headers to the ADNSS service in a simple way. You can copy the below code into the Program.cs file and run it yourself, provided you insert your username and password.

```
using System;
using System.ServiceModel;
using System.ServiceModel.Channels;
using MyDnsClient.DnsServiceTest;

namespace MyDnsClient {
   class Program {
      static void Main(string[] args){

         string nameSpace = "http://groupnbt.com/2010/10/30/Dns/DnsService";
         string userName = "XXXXXX"; // Your username
         string password = "XXXXXX"; // Your password
         string trackingReference = "GETUSER-CFTR65G4FKE";

         Response response = null;
         User user = null;

         // create client object
         DnsServiceClient client = new DnsServiceClient();
         // Get current operation context scope and add headers to the current request
         using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {

OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("UserName",
nameSpace, userName));

OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("Password",
nameSpace, password));
```

```
OperationContext.Current.OutgoingMessageHeaders.Add(MessageHeader.CreateHeader("TrackingRefe
rence", nameSpace, trackingReference));

            // Get user details
            response = client.GetUser(out user, userName);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            if(user != null){
                Console.WriteLine();
                Console.WriteLine("--- User Details ---");
                Console.WriteLine(String.Format("Name: {0}", user.Name));
                Console.WriteLine(String.Format("Username: {0}", user.UserName));
                Console.WriteLine(String.Format("Role: {0}", user.Role));
                Console.WriteLine(String.Format("Email: {0}", user.Email));
            }
            // read the errors if any
        } else if(response.Values != null){
            foreach(string error in response.Values){
                Console.WriteLine(error);
            }
        }
        Console.WriteLine("Press any key to close the program...");
        Console.ReadLine();
    }
  }
}
```

OutPut:

The output should be like below except the details of the user.

StautsCode: 200 StatusMessage: OK TrackingReference: GETUSER-CFTR65G4FKE

--- User Details ---

Name: My Firstname Lastname

Username: XXXXXX

Role: MasterAdministrator

Email: partnerservice@ascio.com

If wrong username or password is provided then you will see the following response:

StautsCode: 401 StatusMessage: Authentication failed

TrackingReference: GETUSER-CFTR65G4FKE

If a username that does not exists in Ascio DNS then the below response will be given.

StautsCode: 414 StatusMessage: User not found

TrackingReference: GETUSER-CFTR65G4FKE

Congratulations, you have now created your first ADNSS program!

4 ADNSS Methods

This section will describe ADNSS methods with examples (one for each method). Each method returns Response object back that tells whether the call succeeded or not. Remember that only the StatusCode 200 is a success and all other values indicate different kind of failures. In every response, unique TechnicalGuid is returned that can be referred to partner service in case you have a specific query for the corresponding request.

4.1 GetRoles

Signature
Response GetRoles(out List<RoleItem> roles);
Usage
Retrieves a list of roles with their corresponding rights
Input Parameters
No input parameter to pass within the request

Responses		
StatusCode	**StatusMessage**	
200	OK	Retrieves a list of roles
401	Authentication failed	

.NET

```
string userName = " XXXXXX";
string password = " XXXXXX";
string trackingReference = "GETROLES-CFTR65G4FKE";
Response response = null;
RoleItem[] roleItems = null;

// create client object
DnsServiceClient client = new DnsServiceClient();
```

```
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)){
            SetHeaders(userName, password, null, trackingReference);


            response = client.GetRoles(out roleItems);
        }
        // close the client
        client.Close();


        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));


        if(response.StatusCode == 200) {
            if(roleItems != null) {
                foreach(RoleItem item in roleItems){
                    Console.WriteLine("--- Role Details ---");
                    Console.WriteLine(String.Format("Role: {0}", item.Role));
                    if(item.Rights != null){
                        Console.WriteLine("Right(s)");
                        foreach(string right in item.Rights){
                            Console.Write("\t");
                            Console.WriteLine(right);
                        }
                    }
                }
            }
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.2 GetUser

Signature
Response GetUser(string userName, out User user);
Usage
Retrieves a specified user details
Input Parameters
userName: The user name to retrieve the details for

Responses		
StatusCode	**StatusMessage**	
200	OK	Retrieves the user details
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to access this user account
414	User not found	

.NET

```
string userName = " XXXXXX";
string password = " XXXXXX ";
string trackingReference = "GETUSER-CFTR65G4FKE";
Response response = null;
User user = null;

// create client object
DnsServiceClient client = new DnsServiceClient();
// Get current operation context scope and add headers to the current request
using(OperationContextScope operationContextScope = new
```

```
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            response = client.GetUser(out user, userName);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));



        if(response.StatusCode == 200) {
            if(user != null) {
                Console.WriteLine("--- User Details ---");
                Console.WriteLine(String.Format("Name: {0}", user.Name));
                Console.WriteLine(String.Format("Username: {0}", user.UserName));
                Console.WriteLine(String.Format("Role: {0}", user.Role));
                Console.WriteLine(String.Format("Email: {0}", user.Email));
            }
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.3 CreateUser

Signature
Response CreateUser(User user);
Usage
Creates a user
Input Parameters
user: The user's details

Responses		
StatusCode	**StatusMessage**	
200	OK	User is created
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation Can only assign role lower than your's
416	Validation failed	FieldName: ErrorMessage e.g. UserName: Required Email: Invalid email address format Role: Does not exist XXXX
203	User already exists	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "CREATEUSER-CFTR65G4FKE";
        Response response = null;
        User user = new User();
        user.Name = "FirstName LastName";
        user.UserName = "newusername";
        user.Role = "StaffAdvanced";
        user.Password = "1234abcd";
        user.Email = "user@email.com";

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            response = client.CreateUser(user);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine(String.Format("User {0} created successfully.", user.UserName));
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
```

```
    }
```

4.4 UpdateUser

Signature
Response UpdateUser(User user);
Usage
Updates user’s Name, Email and Role
Input Parameters
user: The user details

Responses		
StatusCode	**StatusMessage**	
200	OK	User is updated
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation Your role level does not allow you to access this user account Can only assign role lower than your's
409	Operation invalid or not allowed for this object	You cannot update your own role
414	User not found	
416	Validation failed	FieldName: ErrorMessage

		e.g. Email: Invalid email address format Role: Does not exist XXXX

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "UPDATEUSER-CFTR65G4FKE";
        Response response = null;
        User user = null;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            // Get user details
            response = client.GetUser(out user, "newusername");
            if(response.StatusCode == 200) {
                //update email
                user.Email = "user@bestemail.com";
                response = client.UpdateUser(user);
            }

        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));
```

```
if(response.StatusCode == 200) {
    Console.WriteLine(String.Format("User {0} updated successfully.", user.UserName));
    // read the errors if any
} else if(response.Values != null) {
    foreach(string error in response.Values) {
        Console.WriteLine(error);
    }
}
```

4.5 DeleteUser

Signature
Response DeleteUser(string userName);
Usage
Deletes a user
Input Parameters
userName: The username of the user to be deleted

Responses		
StatusCode	**StatusMessage**	
200	OK	Updates the user
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation You cannot delete this user (Own account OR Ascio DNS Partner account)

414	User not found	
409	Operation invalid or not allowed for this object	Deletion failed! User owns zone(s). Delete zone(s) or set new user on zone(s) first!

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "DELETEUSER-CFTR65G4FKE";
        Response response = null;
        string userToBeDeleted = "newusername";

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Delete the user
            response = client.DeleteUser(userToBeDeleted);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine(String.Format("User {0} deleted successfully.", userToBeDeleted));
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
```

```
            Console.WriteLine(error);
        }
    }
```

4.6 SearchUser

Signature
Response SearchUser(SearchUserClause[] searchUserClauses, out string[] userNames);
Usage
Retrieves usernames based on specified search clauses
Input Parameters
searchUserClauses: The search clauses to search usernames

Responses		
StatusCode	**StatusMessage**	
200	OK	The user details is retrieved
400	Parameter value error - Null or invalid value	
401	Authentication failed	
416	Validation failed	FieldName: ErrorMessage e.g. SearchUserClauses: Required at least one search clause SearchUserField: Required Value: Required
204	No results found	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "SEARCHUSER-CFTR65G4FKE";
        Response response = null;
        string[] userNames = null;

        SearchUserClause[] clauses = new SearchUserClause[1];

        SearchUserClause clause = new SearchUserClause();
        clause.SearchUserField = SearchUserField.UserName;
        clause.Value = "*dev*";
        clause.Operator = SearchOperatorType.Like;
        clauses[0] = clause;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            // Get user details
            response = client.SearchUser(out userNames, clauses);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            if(userNames != null) {
                foreach(string name in userNames)
```

```
            Console.WriteLine(String.Format("Username: {0}", name));
        }
        // read the errors if any
    } else if(response.Values != null) {
        foreach(string error in response.Values) {
            Console.WriteLine(error);
        }
    }
```

4.7 ChangePassword

Signature
Response ChangePassword(string userName, string newPassword);
Usage
Changes the password
Input Parameters
userName: The username
newPassword: New password

Responses		
StatusCode	**StatusMessage**	
200	OK	New password is set
400	Parameter value error - Null or invalid value	
401	Authentication failed	
414	User not found	

416	Validation failed	FieldName: ErrorMessage e.g. Password: Length must be between XXX and XXX characters
403	Access denied	Your role level does not allow you to access this user account

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "CHANGEPASSWORD-CFTR65G4FKE";
        Response response = null;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Change password
            response = client.ChangePassword("newusername", "1234abcd12");
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine("Password changed successfully.");
```

```
        // read the errors if any
    } else if(response.Values != null) {
        foreach(string error in response.Values) {
            Console.WriteLine(error);
        }
    }
```

4.8 CreateZone

Signature
Response CreateZone(string zoneName, string owner);
Usage
Creates a zone
Input Parameters
zoneName: The zone name to be created
owner: The username of the user who owns the zone

Responses		
StatusCode	**StatusMessage**	
200	OK	Zone is created
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation Your role level does not allow you to access this user account
414	User not found	

416	Validation failed	e.g. zoneName: Invalid format
201	Zone already exists	Zone already exists in your own account
406	Zone not available	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "CREATEZONE-CFTR65G4FKE";
        Response response = null;
        string zoneName = "mytestzone.com";
        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            // Create zone
            response = client.CreateZone(zoneName, "newusername");
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine(String.Format("Zone {0} created successfully.", zoneName));
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
```

```
            Console.WriteLine(error);
        }
    }
```

4.9 GetZone

Signature
Response GetZone(string zoneName, out Zone zone);
Usage
Retrieves a zone
Input Parameters
zoneName: The zone name to be retrieved

Responses		
StatusCode	**StatusMessage**	
200	OK	Zone is retrieved
400	Parameter value error - Null or invalid value	
401	Authentication failed	
404	Zone not found	

.NET

```
string userName = "XXXXXX";
string password = "XXXXXX";
string trackingReference = "GETZONE-CFTR65G4FKE";
Response response = null;
Zone zone = null;
```

```
        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);


            //Change password
            response = client.GetZone(out zone, "mytestzone.com");
        }
        // close the client
        client.Close();


        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));


        if(response.StatusCode == 200) {
            Console.WriteLine(String.Format("Zone name: {0} ", zone.ZoneName));
            Console.WriteLine(String.Format("Owner: {0}", zone.Owner));
            if(zone.Records != null){
                foreach(Record record in zone.Records){
                  Console.WriteLine(String.Format("Type: {0} Source: {1} Target: {2}", record.GetType().Name
, record.Source, record.Target)); // GetType() is a .NET function. To cast to a specific type, see supported
types in section 5.20
                }
            }
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.10 GetZoneLog

Signature
Response GetZoneLog(string zoneName, out List<ZoneLogEntry> zoneLogEntries);
Usage
Retrieves a list of zone log entries. A zone log entry is added when a zone is deleted, the soa record is updated or any other record is deleted or updated. The log contains information of which user took the action. It will also contain the values of the record as they were before the action was taken.
Input Parameters
zoneName: The zone name to be retrieved

Responses		
StatusCode	**StatusMessage**	
200	OK	List of zone logs is retrieved
400	Parameter value error - Null or invalid value	
401	Authentication failed	
404	Zone not found	

4.11 DeleteZone

Signature
Response DeleteZone(string zoneName);
Usage
Deletes a zone
Input Parameters
zoneName: The zone name to be deleted

Responses		
StatusCode	**StatusMessage**	
200	OK	Zone is created
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation
404	Zone not found	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "DELETEZONE-CFTR65G4FKE";
        Response response = null;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Delete zone
            response = client.DeleteZone("mytestzone.com");
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));
```

```
        if(response.StatusCode == 200) {
            Console.WriteLine("Zone deleted successfully.");
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.12 SearchZoneNames

Signature
Response SearchZoneNames(SearchZoneClaus[] searchZoneClauses, out string[] zoneNames);
Usage
Searches zone(s)
Input Parameters
searchZoneClauses: The search clauses to search zone names

Responses		
StatusCode	**StatusMessage**	
200	OK	Retrieves zone names that meet search clauses within partner portfolio
400	Parameter value error - Null or invalid value	
401	Authentication failed	
416	Validation failed	FieldName: ErrorMessage e.g. SearchZoneClauses: Required at least one search clause SearchZoneField: Required Value: Required
204	No results found	

.NET

```
string userName = "XXXXXX";
string password = "XXXXXX";
string trackingReference = "GETZONE-CFTR65G4FKE";
Response response = null;
string[] zoneNames = null;

SearchZoneClause[] clauses = new SearchZoneClause[1];

SearchZoneClause clause = new SearchZoneClause();
clause.SearchZoneField = SearchZoneField.ZoneName;
clause.Value = "*.com";
clause.Operator = SearchOperatorType.Like;
```

```
        clauses[0] = clause;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Search zonenames
            response = client.SearchZoneNames(out zoneNames, clauses);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            if(zoneNames != null) {
                foreach(string name in zoneNames) {
                    Console.WriteLine(name);
                }
            }
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.13 SearchZone

Signature

Response SearchZone(SearchZoneClaus[] searchZoneClauses, ZoneInfoLevel zoneInfoLevel, out Zone[] zones);
Usage
Searches and retrieves zone(s) with Basic, Partial or Full information
Input Parameters
searchZoneClauses: The search clauses to search zone names
zoneInfoLevel: The information level to retrieve

Responses		
StatusCode	StatusMessage	
200	OK	Retrieves zone(s) that meet search clauses within partner portfolio
400	Parameter value error - Null or invalid value	
401	Authentication failed	
416	Validation failed	FieldName: ErrorMessage e.g. SearchZoneClauses: Required at least one search clause SearchZoneField: Required Value: Required
204	No results found	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "GETZONE-CFTR65G4FKE";
        Response response = null;
        Zone[] zones = null;

        SearchZoneClause[] clauses = new SearchZoneClause[1];

        SearchZoneClause clause = new SearchZoneClause();
        clause.SearchZoneField = SearchZoneField.ZoneName;
        clause.Value = "*zone.com";
        clause.Operator = SearchOperatorType.Like;
        clauses[0] = clause;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Search zones
            response = client.SearchZone(out zones, clauses, ZoneInfoLevel.Partial);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            if(zones != null) {
                foreach(Zone zone in zones) {
```

```
                Console.WriteLine(String.Format("Zone name: {0} ", zone.ZoneName));
                Console.WriteLine(String.Format("Owner: {0}", zone.Owner));
                if(zone.Records != null) {
                    foreach(Record record in zone.Records) {
                        Console.WriteLine(String.Format("Type: {0} Source: {1} Target: {2}",
record.GetType().Name, record.Source, record.Target)); // GetType() is a .NET function. To cast to a
specific type, see supported types in section 5.20
                    }
                }
            }
        }
        // read the errors if any
    } else if(response.Values != null) {
        foreach(string error in response.Values) {
            Console.WriteLine(error);
        }
    }
```

Also see “SearchZoneClause” and “ZoneInfoLevel” for additional examples of searching zones.

4.14 SetZoneOwner

Signature
Response SetZoneOwner(string zoneName, string owner);
Usage
Sets an owner to a zone
Input Parameters
zoneName: The zone name to set the owner for
Owner: The username

Responses		
StatusCode	**StatusMessage**	
200	OK	Owner to a zone is set
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation
404	Zone not found	
414	User not found	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "SETZONEOWNER-CFTR65G4FKE";
        Response response = null;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Set zone owner
            response = client.SetZoneOwner("mytestzone.com", "newOwnerUserName");
        }
        // close the client
        client.Close();

        Console.WriteLine();
```

```
Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
        response.StatusCode, response.StatusMessage, response.TrackingReference));

if(response.StatusCode == 200) {
    Console.WriteLine("New owner is set");
    // read the errors if any
} else if(response.Values != null) {
    foreach(string error in response.Values) {
        Console.WriteLine(error);
    }
}
```

4.15 SetZoneStatus

Signature
Response SetZoneStatus(string zoneName, ZoneStatus status);
Usage
Sets zone status
Input Parameters
zoneName: The zone name to set the status for
status: The zone status

Responses		
StatusCode	**StatusMessage**	
200	OK	Zone status is set
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation

404	Zone not found	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "SETZONESTATUS-CFTR65G4FKE";
        Response response = null;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Set zone status
            response = client.SetZoneStatus("mytestzone1.com", ZoneStatus.Disabled);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine("New status is set");
            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```


4.16 CreateRecord

Signature
Response CreateRecord(string zoneName, Record record, out int recordId);
Usage
Created a record for a zone and provides the newly created record id.
Input Parameters
zoneName: The zone name to create the record for
record: The record to be created

Responses		
StatusCode	**StatusMessage**	
200	OK	Record is created
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation
404	Zone not found	
416	Validation failed	FieldName: ErrorMessage e.g. Target: Invalid IP address format Target: Invalid HTTP url format Source: Required Source: Length must be between

		1 and 255 characters PrimaryNameServer: Invalid Name Server format HostmasterEmail: Invalid email address format TTL: Invalid value
202	Record already exists	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "CREATERECORD-CFTR65G4FKE";
        Response response = null;
        A aRecord = new A();
        aRecord.Source = "mytestzone1.com";
        aRecord.Target = "190.90.10.10";
        aRecord.TTL = 60;
        int recordId = 0;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            //Create record
            response = client.CreateRecord(out recordId, "mytestzone1.com", aRecord);
        }
        // close the client
        client.Close();
```

```
        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine(String.Format("Record created: {0}", recordId));

            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.17 UpdateRecord

Signature
Response UpdateRecord(Record record);
Usage
Updates a record
Input Parameters
record: The record object to update

Responses		
StatusCode	**StatusMessage**	
200	OK	Record is updated
400	Parameter value error - Null or invalid value	
401	Authentication failed	
403	Access denied	Your role level does not allow

		you to do this operation
404	Zone not found	
405	Record not found	
416	Validation failed	FieldName: ErrorMessage e.g. Target: Invalid IP address format Target: Invalid HTTP url format Source: Required Source: Length must be between 1 and 255 characters PrimaryNameServer: Invalid Name Server format HostmasterEmail: Invalid email address format TTL: Invalid value

.NET

```
string userName = "XXXXXX";
string password = "XXXXXX";
string trackingReference = "UPDATERECORD-CFTR65G4FKE";
Response response = null;
Zone zone = null;

// create client object
DnsServiceClient client = new DnsServiceClient();
// Get current operation context scope and add headers to the current request
using(OperationContextScope operationContextScope = new
```

```
OperationContextScope(client.InnerChannel)){
            SetHeaders(userName, password, null, trackingReference);

            response = client.GetZone(out zone, "mytestzone1.com");
            Console.WriteLine("--- Get Zone Response ---");
            Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                                    response.StatusCode, response.StatusMessage,
response.TrackingReference));

            if(response.StatusCode == 200){
               if(zone != null){
                  if(zone.Records != null){
                    A aRecord = zone.Records.First(r => r.Target == "190.90.10.10" && r is A) as A;
                     if(aRecord != null){
                        aRecord.Target = "190.89.20.20"; // update Target
                        aRecord.TTL = 3600; // update TTL
                        response = client.UpdateRecord(aRecord);
                        Console.WriteLine("--- Record Updated Response ---");
                        Console.WriteLine(
                           String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                                    response.StatusCode, response.StatusMessage,
response.TrackingReference));

                        if(response.Values != null){
                           foreach(string error in response.Values){
                              Console.WriteLine(error);
                           }
                        }
                     }
                  }
               }
            }
            // close the client
            client.Close();

        }
```

4.18 DeleteRecord

Signature
Response DeleteRecord(int recordId);
Usage
Deletes a record
Input Parameters
recordId: The id of the record to be deleted

Responses		
StatusCode	**StatusMessage**	
200	OK	Record is deleted
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation
405	Record not found	

.NET

```
string userName = "XXXXXX";
string password = "XXXXXX";
string trackingReference = "DELETERECORD-CFTR65G4FKE";
Response response = null;
Zone zone = null;

// create client object
DnsServiceClient client = new DnsServiceClient();
// Get current operation context scope and add headers to the current request
using(OperationContextScope operationContextScope = new
```

```
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);

            response = client.GetZone(out zone, "mytestzone1.com");
            Console.WriteLine("--- Get Zone Response ---");
            Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                                    response.StatusCode, response.StatusMessage,
response.TrackingReference));



            if(response.StatusCode == 200) {
               if(zone != null) {
                  if(zone.Records != null) {
                     A aRecord = zone.Records.First(r => r.Target == "190.89.20.20" && r is A) as A;
                     if(aRecord != null) {
                        response = client.DeleteRecord(aRecord.Id);
                        Console.WriteLine("--- Record Delete Response ---");
                        Console.WriteLine(
                           String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                                    response.StatusCode, response.StatusMessage,
response.TrackingReference));

                        if(response.Values != null) {
                           foreach(string error in response.Values) {
                              Console.WriteLine(error);
                           }
                        }
                     }
                  }
               }
            }
            // close the client
            client.Close();

         }
```

4.19 GetRecord

Signature
Response GetRecord(int recordId, out Record record);
Usage
Retrievs a record
Input Parameters
recordId: The id of the record to be retrieved

Responses		
StatusCode	**StatusMessage**	
200	OK	Record is deleted
401	Authentication failed	
403	Access denied	Your role level does not allow you to do this operation
405	Record not found	

.NET

```
        string userName = "XXXXXX";
        string password = "XXXXXX";
        string trackingReference = "GETRECORD-CFTR65G4FKE";
        Response response = null;
        Record record = null;

        // create client object
        DnsServiceClient client = new DnsServiceClient();
        // Get current operation context scope and add headers to the current request
        using(OperationContextScope operationContextScope = new
OperationContextScope(client.InnerChannel)) {
            SetHeaders(userName, password, null, trackingReference);
```

```
            //Get record
            response = client.GetRecord(out record, 12345);
        }
        // close the client
        client.Close();

        Console.WriteLine();
        Console.WriteLine(String.Format("StautsCode: {0} StatusMessage: {1} TrackingReference: {2}",
                    response.StatusCode, response.StatusMessage, response.TrackingReference));

        if(response.StatusCode == 200) {
            Console.WriteLine(String.Format("Record found. Source: {0} Target:{1} Type: {2}",
record.Source, record.Target, record.GetType().Name)); // GetType() is .Net function. To cast to a specific
type, see supported types in section 5.20

            // read the errors if any
        } else if(response.Values != null) {
            foreach(string error in response.Values) {
                Console.WriteLine(error);
            }
        }
```

4.20 ADNSS Internal Failure Response by Each Method

We have tried hard to make it possible that each valid request should be succeeded but if still something internally fails then following response is sent by the ADNSS.

StatusCode	**StatusMessage**		
554	Transaction failed		

5 ADNSS Objects

5.1 Response

Field	Type	Description	Example Value
StatusCode	Integer	Request's result code	200
StatusMessage	String	Request's result message	OK
TrackingReference	String	Value provided by the client to track request	
TechnicalGuid	String	Guid returned by ADNSS in each response	
Values	ArrayOfString	Errors if any	

5.2 User

Field	Type	Description	Example Value
UserName	String	Unique user name Length must be between 4 and 65 characters	john1234 john1234@ascio.com
Password	String	Password, alphanumeric Length must be between 4 and 32 characters	John_12E4
Email	String	Length must be between 6 and 65 characters	John.Smith@ascio.com
Role	String	One of the already defined roles	CustomerAdvanced

Name	String	Name of the user	John Smith
CreatedDate	DateTime	Date and Time when user record was created	
UpdatedDate	DateTime	Date and Time when user record was last updated	

5.3 Zone

Field	Type	Description	Example Value
ZoneName	String	Zone name Length must be between 4 and 255 characters	ascio.com
CreatedDate	DateTime	Date and Time when zone was created	
Owner	String	User name of the user who owns zone	john1234
Records	ArrayOfRecord	Records	e.g. SOA, A etc

5.4 ZoneLogEntry

Field	Type	Description	Example Value
ZoneName	String	Zone name	
ActionDate	DateTime	Date and Time when action was taken	
ActionBy	String	User name of the user who took the action	
Record	Record	A copy of the record before the action was taken.	

5.5 Record

Record is an abstract class.

5.6 SOA

The SOA defines global parameters for the zone. There is only one SOA record allowed in a zone file.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	ascio.com
Target	String	This field is ignored for SOA record	
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	
UpdatedDate	DateTime		
Serial	long		
PrimaryNameServer	String	Length must be between 6 and 65 characters	ns1.ascio.com
HostmasterEmail	String	Email address of the person responsible for this zone. Length must be between 6 and 65 characters	hostmaster@ascio.com
Refresh	Integer	Indicates the time when the slave will try to refresh the zone from the master. Value must be between 0 and 2147483647	
SerialUsage	Integer		0 or 1
Retry	Integer	Defines the time between retries if the	

		slave (secondary) fails to contact the master when refresh (above) has expired. Value must be between 0 and 2147483647	
Expire	Integer	Indicates when the zone data is no longer authoritative. Used by Slave or (Secondary) servers only. Value must be between 0 and 2147483647	

5.7 MailForward

Forwards e-mails to another address.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	*@ascio.net
Target	String		mail@ascio.com
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	
UpdatedDate	DateTime		
Serial	long		

5.8 NS

A NS record (name server record) delegates a DNS zone to use the given authoritative name servers.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	ascio.com

Target	String	Namserver's hostname	ns1.ascio.com
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	
UpdatedDate	DateTime		
Serial	long		

5.9 A

An A record (address record) is used to map hostnames (Source) to an ip address (Target).

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	ascio.com
Target	String		192.0.5.6
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	60
UpdatedDate	DateTime		
Serial	long		

5.10 AAAA

AAAA record (IPv6 address record) is used to map hostnames (Source) to a 128-bit IPv6 address (Target).

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	ascio.com

Target	String		2001:0db8:85a3:0000:0000:8a2e:0370:7334
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	60
UpdatedDate	DateTime		
Serial	long		

5.11 CNAME

A CName record (Canonical name record) is an alias of one name to another. The DNS lookup will continue by retrying the lookup with the new name.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	admin.ascio.com
Target	String		administrator.ascio.com
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	60
UpdatedDate	DateTime		
Serial	long		

5.12 PTR

A PTR record (pointer record) points to a canonical name. Unlike a CNAME, DNS processing does NOT proceed, just the name is returned. The most common use is for implementing reverse DNS lookups.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	192.0.5.6.IN-ADDR.ARPA

Target	String		www.ascio.com
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	
UpdatedDate	DateTime		
Serial	long		

5.13 TXT

Provides the ability to associate some arbirary and unformatted text with a host or other name. The TXT record is used to define the “Sender Policy Framework” (SPF) and “DomainKeys Signed Email” (DKIM), information records which may be used to validate legitimate email sources from a domain.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	ascio.com
Target	String		v=spf1 -all
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	60
UpdatedDate	DateTime		
Serial	long		

5.14 MX

A Mx record (mail exchange record) maps a domain name to a list of message transfer agents for that domain.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	ascio.com
Target	String		mail1.ascio.com

TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	60
UpdatedDate	DateTime		
Serial	long		
Priority	Integer	Value between 0 and 65535.	10

5.15 SRV

A SRV record (service locator record) is a generalized service location record, used for newer protocols instead of creating protocol-specific records such as MX.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	_http._tcp.ascio.com
Target	String		www.ascio.com
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	60
UpdatedDate	DateTime		
Serial	long		
Priority	Integer	Value must be between 0 and 65535	0
Weight	Integer		5
Port	Integer		80

5.16 WebForward

Redirects web requests to a different url.

Field	Type	Description	Example Value
Id	long	Id of the record	
Source	String	Zone name	*.ascio.biz
Target	String		http://dns.ascio.com
TTL	Integer	Represents Time to live (NS cache refresh time) for a record Range: 0 to 172800 seconds	
UpdatedDate	DateTime		
Serial	long		
RedirectionType	RedirectionType	Web forward Redirection type	Temporary, Permanent Or Frame

5.17 SearchZoneClause

Using the SearchZoneNames or SearchZone functions requires the user to specify an array of SearchZoneClause. The SearchClause describes what field to use in the search, what kind of operator to use and the value to match.

Adding more than one SearchClause when doing a search will result in an “and” operator between the search clauses as long as the SearchZoneField values are different. Adding more than one SearchClause with the same SearchZoneField value will result in an “or” operator for the search clauses with the same field value (see examples below).

Field	Type	Description	Example Value
SearchZoneField	SearchZoneField	The field to use to create a search criterion. SearchZoneField is an enum, described below.	ZoneName Owner Source Target RecordType CreatedDate TTL
Operator	SearchOperatorType	What kind of search operator to use.	Is

		SearchOperatorType is an enum described below.	IsNot Like NotLike LessThan GreaterThan
Value	String	The value to search for. Wildcards * (multiple character wildcard) or _ (single character wildcard) may be used.	ascio.com ascio.* _scio.com

Examples:

To search for the zone “ascio.com”:

```
SearchZoneClause searchClause = new SearchZoneClause();
searchClause .SearchZoneField = SearchZoneField.ZoneName;
searchClause .Operator = SearchOperatorType.Is;
searchClause .Value = “ascio.com”;
```

To search for zones starting ending with “ascio.com”:

```
SearchZoneClause searchClause = new SearchZoneClause();
searchClause .SearchZoneField = SearchZoneField.ZoneName;
searchClause .Operator = SearchOperatorType.Like;
searchClause .Value = “*ascio.com”;
```

The following will search for all zones where the owner is “subuser@ascio.com” *and* the zone name is “ascio.com” *or* “ascio.net”:

```
SearchZoneClause[] searchClauses = new SearchZoneClause[2];

SearchZoneClause searchClause = new SearchZoneClause();
searchClause .SearchZoneField = SearchZoneField.Owner;
searchClause .Operator = SearchOperatorType.Is;
searchClause .Value = “subuser@ascio.com”;
searchClauses[0] = searchClause;

searchClause = new SearchZoneClause();
searchClause .SearchZoneField = SearchZoneField.ZoneName;
searchClause .Operator = SearchOperatorType.Is;
searchClause .Value = “ascio.com”;
searchClauses[1] = searchClause;
```

```
searchClause = new SearchZoneClause();
searchClause .SearchZoneField = SearchZoneField.ZoneName;
searchClause .Operator = SearchOperatorType.Is;
searchClause .Value = “ascio.net”;
searchClauses[2] = searchClause;
```

5.18 SearchUserClause

Using SearchUser function requires the user to submit an array of SearchUserClause.

Field	Type	Description	Example Value
SearchUserField	SearchUserField	The field to use to create a search criterion. SearchUserField is an enum, described below.	UserName RoleType Email
Operator	SearchOperatorType	What kind of search operator to use. SearchOperatorType is an enum described below.	Is IsNot Like NotLike LessThan GreaterThan
Value	String	The value to search for. Wildcards * (multiple character wildcard) or _ (single character wildcard) may be used.	subuser@ascio.com *@ascio.com

5.19 Enums

5.19.1 RedirectionType

The RedirectionType enum is used when creating a WebForward record, to determine what kind of redirection to use.

Field	Description
Permanent	A permanent redirection is also called a “301” redirect. Permanent 301 redirects are just as they sound. They are permanent redirects from an old URL to a new one. These redirects tell the search engines that the old location is to be removed from their index and replaced with the new location.

	Using 301 redirects is the most search engine friendly way to redirect traffic and engines.
Temporary	A temporary redirect is also called a “302” redirect. Temporary 302 redirects are also as they sound; temporary. Here you are telling the search engines to read and use the content on the new page, but to keep checking the original URL first as it will ultimately be reestablished.
Frame	Redirection is made within a frame. The user will not see the redirection and the web browser will still show the address the user typed in, in the address field.

5.19.2 SearchZoneField

Field	Description
ZoneName	The zone name.
Owner	The owner of the zone.
Source	The source of a record in the zone.
Target	The target of a record in the zone. The target has different meaning for different record types, for an A record the target is an ip-address, but for a webforward record it is a url. See the documentation for each record type.
RecordType	The type of record to search for. See types in 5.20.
CreatedDate	The date the zone was created.
TTL	The record Time To Live.

5.19.3 ZoneInfoLevel

The ZoneInfoLevel is used in SearchZone to set how much information you want to retrieve.

Field	Description
Basic	Making a zone search with ZoneInfoLevel.Basic will return a list of zones matching the search clauses. Each zone in the list will contain information about the zone name, the owner and the create date, but not with the list of records.
Full	Making a zone search with ZoneInfoLevel.Full will return a list of zones matching the search clauses. The list will be populated with the zone name, the owner and the create date, but also with a list of all

	records belonging to the zone. For instance, making a full search where the record target should match a certain ip, will give you a list of zones that has record(s) matching the criteria, but each zone will contain a full list of its records.
Partial	Making a zone search with ZoneLevelInfo.Partial will return a list of zones matching the search clauses. The list will be populated with zone name, owner and zone create date. It will also contain a list of records, but only the records that match. For instance, making a partial search where the record target should match a certain ip, will give you a list of zones that has record(s) matching the criteria, all containing a list of the record that matches.

5.19.4 SearchOperatorType

Field	Description
Is	Equal to.
IsNot	Not equal to.
Like	Like is used when doing wildcard searches. Use * for a multiple character wildcard and _ for a single character wildcard.
NotLike	Not like is also used together with wildcards.
LessThan	Less than. Only applicable to CreateDate and TTL.
GreaterThan	Greater than. Only applicable to CreateDate and TTL.

5.19.5 SearchUserField

Field	Description
UserName	The user name of the user (login user name).
RoleType	The access role type of the user. See “Roles level and description”.
Email	The email address of the user.

5.20 Record types derive from Record class

As described in section 5.3 that a Zone object has a Records property that may contain records of multiple types. All these records derive from Record class. To find the record type of a record, the record object must be casted or compared to a specific type. Following record types are supported by ADNSS:

- **A**
- **AAAA**
- **CNAME**
- **MailForward**
- **MX**
- **NS**
- **PTR**
- **SRV**
- **SOA**
- **TXT**
- **WebForward**