# How to create a SMTP plugin for ArGoSoft Mail Server, .NET edition (AMS .NET edition) using Visual Studio 2005

## *About SMTP plugins for AMS .NET edition*

Plugins should be placed in .NET assemblies. One assembly may contain multiple plugins. Each plugin is a class, which implements *ISmtpPlugin* interface, declared in the *Argosoft.Plugins* assembly, shipped with mail server. Plugins are loaded using reflection at runtime, during each SMTP connection.

*Argosoft.Plugins* assembly declares an enumeration *PostProcessResult*:

```
public enum PostProcessResult
{
   Accept, Reject, Ignore
}
```

Also, four delegates:

```
public delegate void AfterConnectDelegate(object sender,
  string ipAddress, ref bool reject, ref string reply);

public delegate void AfterMailFromDelegate(object sender,
  string ipAddress, string mailFrom, ref bool reject,
  ref string reply);

public delegate void AfterRcptToDelegate(object sender,
  string ipAddress, string rcptTo, bool isLocal, ref bool reject,
  ref string reply);

public delegate void PostProcessDelegate(object sender,
  string ipAddress, ref string mailFrom,
  ref List<string> rcptTo, ref StringBuilder msgData,
  ref PostProcessResult postProcessResult, ref string reply);
```

They correspond to four different phases of SMTP connection, and are types of four event handlers, exposed by *ISmtpPlugin* interface.

```
public interface ISmtpPlugin
{
    AfterConnectDelegate AfterConnect { get; set; }
    AfterMailFromDelegate AfterMailFrom { get; set; }
    AfterRcptToDelegate AfterRcptTo { get; set; }
    PostProcessDelegate PostProcess { get; set; }
    string Description { get; }
}
```

If any of above event handlers is not used, an implementation must return NULL. If server has multiple plugins, events are called in order listed in the plugins.xml file (see below). If at least one handler rejects connection, further plugins are not called.

## AfterConnect event handler

As soon as connection is established, server calls AfterConnect event handlers, before sending a greeting, but after checking an IP address with Auto Lockout (see Security – Auto Lockout in the mail server).

Parameters for *AfterConnect* event handler are as follows:

- object sender - is an instance of *SmtpProcessor* class, which is not documented here;
- string ipAddress – string representation of an IP address of connecting MTA;
- bool reject – a result, returned by plugin. If true, the server will send back a reply, specified in a reply parameter (see below);
- string reply – a error message sent by server. It must start with an appropriate SMTP error code, and contain an explanation, for example: "501 Your IP Address has been rejected" (without double quotes). If reject parameter is false, then reply parameter is ignored.

## AfterMailFrom event handler

Is called after MAIL FROM SMTP command is transmitted, but before the server specifies sender domain name (if enabled).

Parameters for AfterMailFrom event handler are as follows:

- object sender - is an instance of *SmtpProcessor* class, which is not documented here;
- string ipAddress – string representation of an IP address of connecting MTA;
- string mailFrom – an email address, transmitted with MAIL FROM command;
- bool reject – a result, returned by plugin. If true, the server will send back a reply, specified in a reply parameter (see below);
- string reply – a error message sent by server. It must start with an appropriate SMTP error code, and contain an explanation, for example: "501 Your Address has been rejected" (without double quotes). If reject parameter is false, then reply parameter is ignored.

## AfterRcptTo event handler

Is called after each RCPT TO SMTP command.

Parameters for AfterRcptTo event handler are as follows:

- object sender - is an instance of *SmtpProcessor* class, which is not documented here;
- string ipAddress – string representation of an IP address of connecting MTA;
- string rcptTo – an email address, transmitted with RCPT TO command;
- bool local – true if recipient is local;
- bool reject – a result, returned by plugin. If true, the server will send back a reply, specified in a reply parameter (see below);
- string reply – a error message sent by server. It must start with an appropriate SMTP error code, and contain an explanation, for example: "501 Recipient has been rejected" (without double quotes). If reject parameter is false, then reply parameter is ignored.

## PostProcess event handler

This event handler is, perhaps, the most complex among others. It is called after a content of email is transmitted to the server, and delivering MTA expects a confirmation reply from server.

Parameters for PostProcess event handler are as follows:

- object sender - is an instance of *SmtpProcessor* class, which is not documented here;
- string ipAddress – string representation of an IP address of connecting MTA;
- string mailFrom – an email address, transmitted with MAIL FROM command;
- List<string> rcptTo – an entire list of addresses, transmitted by MTA, and not rejected by mail server (or plugins). This list can be altered by PostProcess event handler, server will accept the changes, and deliver mail accordingly (of course, depending on the value of postProcessResult parameter, see below);
- StringBuilder msgData – it is a rfc822 message, exactly as it was accepted by server. The content of this information also can be altered by the event handler;
- PostProcessResult postProcessResult – it is the out parameter. If it's value is PostProcessResult.Accept, then message will be accepted, and control will be passed to the server, if it is PostProcessResult.Ignore, then server will assume, that processing for the message has been completely done by the server. It will notify the delivering MTA, that email has been accepted, but will not do any further work for delivering mail to recipients. If the value of reply is

PostProcessResult.Reject, then server will send back an error message, contained in the reply parameter (see below);

- string reply – a error message sent by server. It must start with an appropriate SMTP error code, and contain an explanation, for example: "501 Your message has been rejected" (without double quotes). This parameter will be used only when postProcessResult (see above) is PostProcessResult.Reject;

## Sample SMTP Plugin

To begin with, let's create a "skeleton" of our future SMTP plugin.

Launch Visual Studio 2005, create a class library, name it *TestPlugin*, add a reference to *Argosoft.Plugins.dll* assembly (it is located in the root directory of mail server installation), and add a class *SenderDomainChecker*, add a reference to Argosoft.Plugins assembly (it is located  Content of a SMTP plugin, which does nothing, is below:

```
using System;
using System.Collections.Generic;
using System.Text;
using Argosoft.Plugins;

namespace TestPlugin
{
    public class SenderDomainChecker : ISmtpPlugin
    {
        public AfterConnectDelegate AfterConnect
        {
            get { return null; }
            set { }
        }
        public AfterMailFromDelegate AfterMailFrom
        {
            get { return null; }
            set { }
        }
        public AfterRcptToDelegate AfterRcptTo
        {
            get { return null; }
            set { }
        }
        public PostProcessDelegate PostProcess
        {
            get { return null; }
            set { }
        }
        public string Description
        {
            get { return "Test Description"; }
        }
    }
}
```

Now, let's create a plugin, which checks whether the domain name of sender exists. To do it, add to the above class an event handler for AfterMailFrom delegate, and initialize the handler. Also, we will have to add *System.Net* and *System.Net.Sockets* to the "using" section. Final code would look like this:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using Argosoft.Plugins;

namespace TestPlugin
{
    public class SenderDomainChecker : ISmtpPlugin
    {
        public AfterConnectDelegate AfterConnect
        {
            get { return null; }
            set { }
        }
        private AfterMailFromDelegate afterMailFrom = new
AfterMailFromDelegate(CheckDomain);
        public AfterMailFromDelegate AfterMailFrom
        {
            get { return afterMailFrom; }
            set { }
        }
        public AfterRcptToDelegate AfterRcptTo
        {
            get { return null; }
            set { }
        }
        public PostProcessDelegate PostProcess
        {
            get { return null; }
            set { }
        }
        public string Description
        {
            get { return "Checks if domain name of sender is valid"; }
        }
        private static string PickDomain(string address)
        {
            int i = address.IndexOf('@');
            if (i > -1)
                return address.Substring(i + 1);
            else
                return "";
        }
        public static void CheckDomain(object sender, string ipAddress,
string mailFrom,
            ref bool reject, ref string reply)
        {
            reject = false;
            reply = "";
```

```
            string domain = PickDomain(mailFrom);
            if (domain != "")
            {
                try
                {
                    IPAddress[] ipAddresses =
Dns.GetHostAddresses(domain);
                    return;
                }
                catch (SocketException)
                {
                    reject = true;
                    reply = String.Format("501 Sender domain {0}
appears to be invalid", domain);
                }
                catch { }
            }
        }
    }
}
```

Now, we can build our assembly, move resulting DLL file into the mail server root directory, and add it to the list of plugins Security – SMTP Plugins and use it.

To add it to the list of plugins, run the mail server UI, go to Security – SMTP Plugins, and add assembly name and class name: