```
-- ============================================
-- Author:      Vittorio Polizzi
--              Questo script è stato scaricato da http://vpolizzi.wordpress.com
-- Create date: September 04, 2010
-- Description: Script per la creazione delle tabelle e delle versioni modificare
--              delle stored procedure di base per l'esecuzione delle funzionalità
--              Database Mail su Microsoft SQL Azure.
-- ============================================

-- ============================================
-- CREAZIONE DELLE TABELLE
-- ============================================

CREATE TABLE [dbo].[sysmail_account](
        [account_id] [int] IDENTITY(1,1) NOT NULL,
        [name] [sysname] NOT NULL,
        [description] [nvarchar](256) NULL,
        [email_address] [nvarchar](128) NOT NULL,
        [display_name] [nvarchar](128) NULL,
        [replyto_address] [nvarchar](128) NULL,
        [last_mod_datetime] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [SYSMAIL_ACCOUNT_IDMustBeUnique] PRIMARY KEY CLUSTERED
(
        [account_id] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF) ,
 CONSTRAINT [SYSMAIL_ACCOUNT_NameMustBeUnique] UNIQUE NONCLUSTERED
(
        [name] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

ALTER TABLE [dbo].[sysmail_account] ADD  DEFAULT (getdate()) FOR [last_mod_datetime]
GO

ALTER TABLE [dbo].[sysmail_account] ADD  DEFAULT (SYSTEM_USER) FOR [last_mod_user]
GO

CREATE TABLE [dbo].[sysmail_servertype](
        [servertype] [sysname] NOT NULL,
        [is_incoming] [bit] NOT NULL,
        [is_outgoing] [bit] NOT NULL,
        [last_mod_datetime] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [SYSMAIL_SERVERTYPE_TypeMustBeUnique] PRIMARY KEY CLUSTERED
(
        [servertype] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

ALTER TABLE [dbo].[sysmail_servertype] ADD  DEFAULT ((0)) FOR [is_incoming]
GO

ALTER TABLE [dbo].[sysmail_servertype] ADD  DEFAULT ((1)) FOR [is_outgoing]
GO

ALTER TABLE [dbo].[sysmail_servertype] ADD  DEFAULT (getdate()) FOR [last_mod_datetime]
GO

ALTER TABLE [dbo].[sysmail_servertype] ADD  DEFAULT (suser_sname()) FOR [last_mod_user]
GO
```

```
INSERT INTO [dbo].[sysmail_servertype]
(servertype,is_incoming,is_outgoing,last_mod_datetime,last_mod_user)
VALUES
('SMTP',0,1,GETDATE(),suser_sname())
GO

CREATE TABLE [dbo].[sysmail_server](
        [account_id] [int] NOT NULL,
        [servertype] [sysname] NOT NULL,
        [servername] [sysname] NOT NULL,
        [port] [int] NOT NULL,
        [username] [nvarchar](128) NULL,
        [credential_id] [int] NULL,
        [use_default_credentials] [bit] NOT NULL,
        [enable_ssl] [bit] NOT NULL,
        [flags] [int] NOT NULL,
        [last_mod_datetime] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [SYSMAIL_ACCOUNT_AccountServerTypeMustBeUnique] PRIMARY KEY CLUSTERED
(
        [account_id] ASC,
        [servertype] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

ALTER TABLE [dbo].[sysmail_server]  WITH CHECK ADD FOREIGN KEY([account_id])
REFERENCES [dbo].[sysmail_account] ([account_id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[sysmail_server]  WITH CHECK ADD FOREIGN KEY([servertype])
REFERENCES [dbo].[sysmail_servertype] ([servertype])
GO

ALTER TABLE [dbo].[sysmail_server] ADD  DEFAULT ((25)) FOR [port]
GO

ALTER TABLE [dbo].[sysmail_server] ADD  DEFAULT ((0)) FOR [use_default_credentials]
GO

ALTER TABLE [dbo].[sysmail_server] ADD  DEFAULT ((0)) FOR [enable_ssl]
GO

ALTER TABLE [dbo].[sysmail_server] ADD  DEFAULT ((0)) FOR [flags]
GO

ALTER TABLE [dbo].[sysmail_server] ADD  DEFAULT (getdate()) FOR [last_mod_datetime]
GO

ALTER TABLE [dbo].[sysmail_server] ADD  DEFAULT (suser_sname()) FOR [last_mod_user]
GO

CREATE TABLE [dbo].[sysmail_profile](
        [profile_id] [int] IDENTITY(1,1) NOT NULL,
        [name] [sysname] NOT NULL,
        [description] [nvarchar](256) NULL,
        [last_mod_datetime] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [SYSMAIL_PROFILE_IDMustBeUnique] PRIMARY KEY CLUSTERED
(
        [profile_id] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF) ,
 CONSTRAINT [SYSMAIL_PROFILE_NameMustBeUnique] UNIQUE NONCLUSTERED
(
        [name] ASC
```

```
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

ALTER TABLE [dbo].[sysmail_profile] ADD  DEFAULT (getdate()) FOR [last_mod_datetime]
GO

ALTER TABLE [dbo].[sysmail_profile] ADD  DEFAULT (suser_sname()) FOR [last_mod_user]
GO

CREATE TABLE [dbo].[sysmail_profileaccount](
        [profile_id] [int] NOT NULL,
        [account_id] [int] NOT NULL,
        [sequence_number] [int] NULL,
        [last_mod_datetime] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [SYSMAIL_ACCOUNT_ProfileAccountMustBeUnique] PRIMARY KEY CLUSTERED
(
        [profile_id] ASC,
        [account_id] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

ALTER TABLE [dbo].[sysmail_profileaccount]  WITH CHECK ADD FOREIGN KEY([account_id])
REFERENCES [dbo].[sysmail_account] ([account_id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[sysmail_profileaccount] ADD  DEFAULT (getdate()) FOR [last_mod_datetime]
GO

ALTER TABLE [dbo].[sysmail_profileaccount] ADD  DEFAULT (suser_sname()) FOR [last_mod_user]
GO

CREATE TABLE [dbo].[sysmail_principalprofile](
        [profile_id] [int] NOT NULL,
        [principal_sid] [varbinary](85) NOT NULL,
        [is_default] [bit] NOT NULL,
        [last_mod_datetime] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [SYSMAIL_PRINCIPALPROFILE_ProfilePrincipalMustBeUnique] PRIMARY KEY CLUSTERED
(
        [profile_id] ASC,
        [principal_sid] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[sysmail_principalprofile]  WITH CHECK ADD FOREIGN KEY([profile_id])
REFERENCES [dbo].[sysmail_profile] ([profile_id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[sysmail_principalprofile] ADD  DEFAULT ((0)) FOR [is_default]
GO

ALTER TABLE [dbo].[sysmail_principalprofile] ADD  DEFAULT (getdate()) FOR [last_mod_datetime]
GO

ALTER TABLE [dbo].[sysmail_principalprofile] ADD  DEFAULT (suser_sname()) FOR [last_mod_user]
```

```
GO

CREATE TABLE [dbo].[sysmail_mailitems](
        [mailitem_id] [int] IDENTITY(1,1) NOT NULL,
        [profile_id] [int] NOT NULL,
        [recipients] [varchar](max) NULL,
        [copy_recipients] [varchar](max) NULL,
        [blind_copy_recipients] [varchar](max) NULL,
        [subject] [nvarchar](255) NULL,
        [from_address] [varchar](max) NULL,
        [reply_to] [varchar](max) NULL,
        [body] [nvarchar](max) NULL,
        [body_format] [varchar](20) NULL,
        [importance] [varchar](6) NULL,
        [sensitivity] [varchar](12) NULL,
        [file_attachments] [nvarchar](max) NULL,
        [attachment_encoding] [varchar](20) NULL,
        [query] [nvarchar](max) NULL,
        [execute_query_database] [sysname] NULL,
        [attach_query_result_as_file] [bit] NULL,
        [query_result_header] [bit] NULL,
        [query_result_width] [int] NULL,
        [query_result_separator] [char](1) NULL,
        [exclude_query_output] [bit] NULL,
        [append_query_error] [bit] NULL,
        [send_request_date] [datetime] NOT NULL,
        [send_request_user] [sysname] NOT NULL,
        [sent_account_id] [int] NULL,
        [sent_status] [tinyint] NULL,
        [sent_date] [datetime] NULL,
        [last_mod_date] [datetime] NOT NULL,
        [last_mod_user] [sysname] NOT NULL,
 CONSTRAINT [sysmail_mailitems_id_MustBeUnique] PRIMARY KEY CLUSTERED
(
        [mailitem_id] ASC
)WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF)
)

GO

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[sysmail_mailitems]  WITH CHECK ADD  CONSTRAINT
[sysmail_OutMailAttachmentEncodingMustBeValid] CHECK  (([attachment_encoding]='UUENCODE' OR
[attachment_encoding]='BINHEX' OR [attachment_encoding]='S/MIME' OR [attachment_encoding]='MIME'))
GO

ALTER TABLE [dbo].[sysmail_mailitems] CHECK CONSTRAINT [sysmail_OutMailAttachmentEncodingMustBeValid]
GO

ALTER TABLE [dbo].[sysmail_mailitems]  WITH CHECK ADD  CONSTRAINT [sysmail_OutMailImportanceMustBeValid]
CHECK  (([importance]='HIGH' OR [importance]='NORMAL' OR [importance]='LOW'))
GO

ALTER TABLE [dbo].[sysmail_mailitems] CHECK CONSTRAINT [sysmail_OutMailImportanceMustBeValid]
GO

ALTER TABLE [dbo].[sysmail_mailitems]  WITH CHECK ADD  CONSTRAINT
[sysmail_OutMailMustHaveAtleastOneRecipient] CHECK  ((NOT ([recipients] IS NULL AND [copy_recipients] IS
NULL AND [blind_copy_recipients] IS NULL)))
GO

ALTER TABLE [dbo].[sysmail_mailitems] CHECK CONSTRAINT [sysmail_OutMailMustHaveAtleastOneRecipient]
GO

ALTER TABLE [dbo].[sysmail_mailitems]  WITH CHECK ADD  CONSTRAINT [sysmail_OutMailRecipientCannotBeEmpty]
```

```
CHECK  ((((datalength(isnull([recipients],''))+datalength(isnull([copy_recipients],'')))+datalength(isnull
([blind_copy_recipients],'')))<>(0)))
GO

ALTER TABLE [dbo].[sysmail_mailitems] CHECK CONSTRAINT [sysmail_OutMailRecipientCannotBeEmpty]
GO

ALTER TABLE [dbo].[sysmail_mailitems]  WITH CHECK ADD  CONSTRAINT [sysmail_OutMailSensitivityMustBeValid]
CHECK  (([sensitivity]='CONFIDENTIAL' OR [sensitivity]='PRIVATE' OR [sensitivity]='PERSONAL' OR
[sensitivity]='NORMAL'))
GO

ALTER TABLE [dbo].[sysmail_mailitems] CHECK CONSTRAINT [sysmail_OutMailSensitivityMustBeValid]
GO

ALTER TABLE [dbo].[sysmail_mailitems] ADD  DEFAULT (getdate()) FOR [send_request_date]
GO

ALTER TABLE [dbo].[sysmail_mailitems] ADD  DEFAULT (suser_sname()) FOR [send_request_user]
GO

ALTER TABLE [dbo].[sysmail_mailitems] ADD  DEFAULT ((0)) FOR [sent_status]
GO

ALTER TABLE [dbo].[sysmail_mailitems] ADD  DEFAULT (getdate()) FOR [last_mod_date]
GO

ALTER TABLE [dbo].[sysmail_mailitems] ADD  DEFAULT (suser_sname()) FOR [last_mod_user]
GO

CREATE TABLE sysmail_account_credential(
        credential_id int IDENTITY(1,1) NOT NULL,
        username nvarchar(256) NULL,
        cyphertext nvarchar(256) NULL,
 CONSTRAINT [SYSMAIL_ACCOUNT_CREDENTIALIDMustBeUnique] PRIMARY KEY CLUSTERED
        (
                credential_id ASC
        )
WITH (STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF))

GO

-- ================================================
-- CREAZIONE DELLE STORED PROCEDURE E DELLE FUNCTION
-- ================================================


CREATE PROCEDURE [dbo].[sysmail_verify_profile_sp]
   @profile_id int,
   @profile_name sysname,
   @allow_both_nulls bit,
   @allow_id_name_mismatch bit,
   @profileid int OUTPUT
AS
   IF @allow_both_nulls = 0
   BEGIN
      -- at least one parameter must be supplied
      IF (@profile_id IS NULL AND @profile_name IS NULL)
      BEGIN
         RAISERROR(14604, -1, -1, 'profile')
         RETURN(1)
      END
   END

   IF ((@allow_id_name_mismatch = 0) AND (@profile_id IS NOT NULL AND @profile_name IS NOT NULL)) -- use
both parameters
   BEGIN
```

```
      SELECT @profileid = profile_id FROM dbo.sysmail_profile WHERE profile_id=@profile_id AND
name=@profile_name
      IF (@profileid IS NULL) -- id and name do not match
      BEGIN
         RAISERROR(14605, -1, -1, 'profile')
         RETURN(2)
      END
   END
   ELSE IF (@profile_id IS NOT NULL) -- use id
   BEGIN
      SELECT @profileid = profile_id FROM dbo.sysmail_profile WHERE profile_id=@profile_id
      IF (@profileid IS NULL) -- id is invalid
      BEGIN
         RAISERROR(14606, -1, -1, 'profile')
         RETURN(3)
      END
   END
   ELSE IF (@profile_name IS NOT NULL) -- use name
   BEGIN
      SELECT @profileid = profile_id FROM dbo.sysmail_profile WHERE name=@profile_name
      IF (@profileid IS NULL) -- name is invalid
      BEGIN
         RAISERROR(14607, -1, -1, 'profile')
         RETURN(4)
      END
   END
   RETURN(0) -- SUCCESS
GO

CREATE PROCEDURE [dbo].[sysmail_verify_principal_sp]
   @principal_id int,
   @principal_name sysname,
   @allow_both_nulls bit,
   @principal_sid varbinary(85) OUTPUT
AS
   IF @allow_both_nulls = 0
   BEGIN
      -- at least one parameter must be supplied
      IF (@principal_id IS NULL AND @principal_name IS NULL)
      BEGIN
         RAISERROR(14604, -1, -1, 'principal')
         RETURN(1)
      END
   END

   DECLARE @principalid int

   IF (@principal_id IS NOT NULL AND @principal_name IS NOT NULL) -- both parameters supplied
   BEGIN
     SELECT @principalid=principal_id FROM sys.database_principals
            WHERE type in ('U','S','G') AND principal_id = @principal_id AND name = @principal_name

      IF (@principalid IS NULL)
      BEGIN
         RAISERROR(14605, -1, -1, 'principal')
         RETURN(2)
      END
   END
   ELSE IF (@principal_id IS NOT NULL) -- use id
   BEGIN
     SELECT @principalid=principal_id FROM sys.database_principals
            WHERE type in ('U','S','G') AND principal_id = @principal_id

      IF (@principalid IS NULL)
      BEGIN
         RAISERROR(14606, -1, -1, 'principal')
         RETURN(3)
```

```
      END
   END
   ELSE IF (@principal_name IS NOT NULL)  -- use name
   BEGIN
     SELECT @principalid=principal_id FROM sys.database_principals
            WHERE type in ('U','S','G') AND name = @principal_name

      IF (@principalid IS NULL)
      BEGIN
         RAISERROR(14607, -1, -1, 'principal')
         RETURN(4)
      END
   END

   -- populate return variable
   SELECT @principal_sid = dbo.get_principal_sid(@principalid)

   RETURN(0) -- SUCCESS
GO

CREATE PROCEDURE [dbo].[sysmail_verify_accountparams_sp]
   @use_default_credentials bit,
   @mailserver_type sysname      OUTPUT,  -- @mailserver_type must be provided. Usually SMTP
   @username      nvarchar(128)  OUTPUT, -- returns trimmed value, NULL if empty
   @password      nvarchar(128)  OUTPUT  -- returns trimmed value,  NULL if empty
AS
   SET @username = LTRIM(RTRIM(@username))
   SET @password = LTRIM(RTRIM(@password))
   SET @mailserver_type = LTRIM(RTRIM(@mailserver_type))

    IF(@username = N'')         SET @username = NULL
    IF(@password = N'')         SET @password = NULL
    IF(@mailserver_type = N'')  SET @mailserver_type = NULL

   IF(@mailserver_type IS NULL)
   BEGIN
      RAISERROR(14614, -1, -1, @mailserver_type)
      RETURN (1)
   END

   -- default credentials should supercede any explicit credentials passed in
   IF((@use_default_credentials = 1) AND (@username IS NOT NULL))
   BEGIN
      RAISERROR(14666, -1, -1)
      RETURN (1)
   END

   --If a password is specified then @username must be a non empty string
   IF((@password IS NOT NULL) AND (@username IS NULL))
   BEGIN
      RAISERROR(14615, -1, -1)
      RETURN (1)
   END

   RETURN(0) -- SUCCESS
GO

CREATE PROCEDURE [dbo].[sysmail_verify_account_sp]
   @account_id int,
   @account_name sysname,
   @allow_both_nulls bit,
   @allow_id_name_mismatch bit,
   @accountid int OUTPUT
AS
   IF @allow_both_nulls = 0
   BEGIN
      -- at least one parameter must be supplied
```

```
      IF (@account_id IS NULL AND @account_name IS NULL)
      BEGIN
         RAISERROR(14604, -1, -1, 'account')
         RETURN(1)
      END
   END

   IF ((@allow_id_name_mismatch = 0) AND (@account_id IS NOT NULL AND @account_name IS NOT NULL)) -- use
both parameters
   BEGIN
      SELECT @accountid = account_id FROM dbo.sysmail_account WHERE account_id=@account_id AND
name=@account_name
      IF (@accountid IS NULL) -- id and name do not match
      BEGIN
         RAISERROR(14605, -1, -1, 'account')
         RETURN(2)
      END
   END
   ELSE IF (@account_id IS NOT NULL) -- use id
   BEGIN
      SELECT @accountid = account_id FROM dbo.sysmail_account WHERE account_id=@account_id
      IF (@accountid IS NULL) -- id is invalid
      BEGIN
         RAISERROR(14606, -1, -1, 'account')
         RETURN(3)
      END
   END
   ELSE IF (@account_name IS NOT NULL) -- use name
   BEGIN
      SELECT @accountid = account_id FROM dbo.sysmail_account WHERE name=@account_name
      IF (@accountid IS NULL) -- name is invalid
      BEGIN
         RAISERROR(14607, -1, -1, 'account')
         RETURN(4)
      END
   END
   RETURN(0) -- SUCCESS
GO

CREATE PROCEDURE [dbo].[sysmail_create_user_credential_sp]
   @username      nvarchar(128),
   @password      nvarchar(128),
   @credential_id int            OUTPUT
AS
   -- Le porzioni commentate fanno riferimento al codice originale

   SET NOCOUNT ON
   --DECLARE @rc int
   --DECLARE @credential_name UNIQUEIDENTIFIER
   DECLARE @credential_name_as_str varchar(40)
   --DECLARE @sql NVARCHAR(max)

   ---- create a GUID as the name for the credential
   --SET @credential_name = newid()
   SET @credential_name_as_str = convert(varchar(40), @username) --@credential_name)
   --SET @sql = N'CREATE CREDENTIAL [' + @credential_name_as_str
   --         + N'] WITH IDENTITY = ' + QUOTENAME(@username, '''')
   --         + N', SECRET = ' + QUOTENAME(ISNULL(@password, N''), '''')

   --EXEC @rc = sp_executesql @statement = @sql
   --IF(@rc <> 0)
   --   RETURN @rc

   INSERT INTO dbo.sysmail_account_credential (username,cyphertext) VALUES (@username, @password)

   --SELECT @credential_id = credential_id
   --FROM sys.credentials
```

```
   --WHERE name = convert(sysname, @credential_name)

   SELECT @credential_id = credential_id FROM dbo.sysmail_account_credential WHERE credential_id =
@@IDENTITY

    IF(@credential_id IS NULL)
   BEGIN
      RAISERROR(14616, -1, -1, @credential_name_as_str)
      RETURN 1
   END

   RETURN(0)
GO

CREATE PROCEDURE [dbo].[sysmail_add_profileaccount_sp]
   @profile_id int = NULL, -- must provide id or name
   @profile_name sysname = NULL,
   @account_id int = NULL, -- must provide id or name
   @account_name sysname = NULL,
   @sequence_number int -- account with the lowest sequence number is picked as default
AS
   SET NOCOUNT ON

   DECLARE @rc int
   DECLARE @profileid int
   DECLARE @accountid int

   exec @rc = dbo.sysmail_verify_profile_sp @profile_id, @profile_name, 0, 0, @profileid OUTPUT
   IF @rc <> 0
      RETURN(1)

   exec @rc = dbo.sysmail_verify_account_sp @account_id, @account_name, 0, 0, @accountid OUTPUT
   IF @rc <> 0
      RETURN(2)

   -- insert new account record, rely on primary key constraint to error out
   INSERT INTO dbo.sysmail_profileaccount (profile_id,account_id,sequence_number)
   VALUES (@profileid,@accountid,@sequence_number)

   RETURN(0)
GO

CREATE PROCEDURE [dbo].[sysmail_add_profile_sp]
   @profile_name sysname,
   @description nvarchar(256) = NULL,
   @profile_id int = NULL OUTPUT
AS
   SET NOCOUNT ON

   -- insert new profile record, rely on primary key constraint to error out
   INSERT INTO dbo.sysmail_profile (name,description) VALUES (@profile_name, @description)

   -- fetch back profile_id
   SELECT @profile_id = profile_id FROM dbo.sysmail_profile WHERE name = @profile_name

   RETURN(0)
GO

CREATE PROCEDURE [dbo].[sysmail_add_principalprofile_sp]
   @principal_id int = NULL, -- must provide id or name
   @principal_name sysname = NULL,
   @profile_id int = NULL, -- must provide id or name
   @profile_name sysname = NULL,
   @is_default bit
AS
   SET NOCOUNT ON
```

```
   DECLARE @rc int
   DECLARE @principal_sid varbinary(85)
   DECLARE @profileid int

   IF (@principal_id IS NOT NULL AND @principal_id = 0) OR (@principal_name IS NOT NULL AND
@principal_name = N'public')
   BEGIN
      IF (@principal_id IS NOT NULL AND @principal_id <> 0) OR (@principal_name IS NOT NULL AND
@principal_name <> N'public')
      BEGIN
         RAISERROR(14605, -1, -1, 'principal')  -- id and name do not match
      END
      SET @principal_sid = 0x00 -- public
   END
   ELSE
   BEGIN
      exec @rc = dbo.sysmail_verify_principal_sp @principal_id, @principal_name, 0, @principal_sid OUTPUT
      IF @rc <> 0
         RETURN(2)
   END

   exec @rc = dbo.sysmail_verify_profile_sp @profile_id, @profile_name, 0, 0, @profileid OUTPUT
   IF @rc <> 0
      RETURN(3)

   -- insert new account record, rely on primary key constraint to error out
   INSERT INTO dbo.sysmail_principalprofile (principal_sid,profile_id,is_default)
   VALUES (@principal_sid,@profileid,@is_default)

   IF (@is_default IS NOT NULL AND @is_default = 1 )
   BEGIN
      -- a principal can only have one default profile so reset other, if there are any
      UPDATE dbo.sysmail_principalprofile
      SET is_default=0
      WHERE principal_sid = @principal_sid AND profile_id <> @profileid
   END
   RETURN(0)
GO

CREATE PROCEDURE [dbo].[sysmail_add_account_sp]
   @account_name sysname,
   @email_address nvarchar(128),
   @display_name nvarchar(128) = NULL,
   @replyto_address nvarchar(128) = NULL,
   @description nvarchar(256) = NULL,
   @mailserver_name sysname = NULL, -- the following fields are part of server definition
   @mailserver_type sysname = N'SMTP',
   @port int = 25,
   @username nvarchar(128) = NULL,
   @password nvarchar(128) = NULL,
   @use_default_credentials bit = 0,
   @enable_ssl bit = 0,
   @account_id int = NULL OUTPUT
AS
   SET NOCOUNT ON
   DECLARE @rc int
   DECLARE @credential_id int

   EXEC @rc = dbo.sysmail_verify_accountparams_sp
            @use_default_credentials = @use_default_credentials,
            @mailserver_type = @mailserver_type OUTPUT, -- validates and returns trimmed value
            @username = @username OUTPUT, -- returns trimmed value, NULL if empty
            @password = @password OUTPUT  -- returns trimmed value, NULL if empty
   IF(@rc <> 0)
      RETURN (1)

   --transact this in case sysmail_create_user_credential_sp fails
```

```
   BEGIN TRANSACTION

   -- insert new account record, rely on primary key constraint to error out
   INSERT INTO dbo.sysmail_account (name,description,email_address,display_name,replyto_address)
   VALUES (@account_name,@description,@email_address,@display_name,@replyto_address)
   IF (@@ERROR <> 0)
   BEGIN
      ROLLBACK TRANSACTION
      RETURN (2)
   END

   -- fetch back account_id
   SELECT @account_id = account_id FROM dbo.sysmail_account WHERE name = @account_name

   IF (@mailserver_name IS NULL) -- use local server as default
      SELECT @mailserver_name=@@SERVERNAME

   --create a credential in the credential store if a password needs to be stored
   IF(@username IS NOT NULL)
   BEGIN
      EXEC @rc = dbo.sysmail_create_user_credential_sp
                     @username,
                     @password,
                     @credential_id OUTPUT
      IF(@rc <> 0)
      BEGIN
         ROLLBACK TRANSACTION
         RETURN (3)
      END
   END

   INSERT INTO dbo.sysmail_server
(account_id,servertype,servername,port,username,credential_id,use_default_credentials,enable_ssl)
   VALUES
(@account_id,@mailserver_type,@mailserver_name,@port,@username,@credential_id,@use_default_credentials,@enable
   IF (@@ERROR <> 0)
   BEGIN
      ROLLBACK TRANSACTION
      RETURN (4)
   END

   COMMIT TRANSACTION
   RETURN(0)
GO

CREATE FUNCTION [dbo].[get_principal_id] (@sid varbinary(85))
RETURNS int
AS
BEGIN
        DECLARE @id int
        SELECT @id = principal_id FROM sys.database_principals WHERE sid=@sid
        RETURN @id
END
GO

CREATE FUNCTION [dbo].[get_principal_sid] (@id int)
RETURNS varbinary(85)
AS
BEGIN
        DECLARE @sid varbinary(85)
        SELECT @sid = sid  FROM sys.database_principals WHERE principal_id=@id
        RETURN @sid
END
GO

CREATE PROCEDURE [dbo].[sp_send_dbmail]
   @profile_name               sysname    = NULL,
```

```
   @recipients                 VARCHAR(MAX)  = NULL,
   @copy_recipients            VARCHAR(MAX)  = NULL,
   @blind_copy_recipients      VARCHAR(MAX)  = NULL,
   @subject                    NVARCHAR(255) = NULL,
   @body                       NVARCHAR(MAX) = NULL,
   @body_format                VARCHAR(20)   = NULL,
   @importance                 VARCHAR(6)    = 'NORMAL',
   @sensitivity                VARCHAR(12)   = 'NORMAL',
   @file_attachments           NVARCHAR(MAX) = NULL,
   @query                      NVARCHAR(MAX) = NULL,
   @execute_query_database     sysname       = NULL,
   @attach_query_result_as_file BIT          = 0,
   @query_attachment_filename  NVARCHAR(260) = NULL,
   @query_result_header        BIT           = 1,
   @query_result_width         INT           = 256,
   @query_result_separator     CHAR(1)       = ' ',
   @exclude_query_output       BIT           = 0,
   @append_query_error         BIT           = 0,
   @query_no_truncate          BIT           = 0,
   @query_result_no_padding    BIT           = 0,
   @mailitem_id                INT           = NULL OUTPUT,
   @from_address               VARCHAR(max)  = NULL,
   @reply_to                   VARCHAR(max)  = NULL
  --WITH EXECUTE AS 'dbo'
AS
BEGIN
    SET NOCOUNT ON

    -- And make sure ARITHABORT is on. This is the default for yukon DB's
    SET ARITHABORT ON

    --Declare variables used by the procedure internally
    DECLARE @profile_id         INT,
            @temp_table_uid     uniqueidentifier,
            @sendmailxml        VARCHAR(max),
            @CR_str             NVARCHAR(2),
            @localmessage       NVARCHAR(255),
            @QueryResultsExist  INT,
            @AttachmentsExist   INT,
            @RetErrorMsg        NVARCHAR(4000), --Impose a limit on the error message length to avoid
memory abuse
            @rc                 INT,
            @procName           sysname,
            @trancountSave      INT,
            @tranStartedBool    INT,
            @is_sysadmin        BIT,
            @send_request_user  sysname,
            @database_user_id   INT,
            @sid                varbinary(85)

    -- Initialize
    SELECT  @rc                 = 0,
            @QueryResultsExist  = 0,
            @AttachmentsExist   = 0,
            @temp_table_uid     = NEWID(),
            @procName           = OBJECT_NAME(@@PROCID),
            @tranStartedBool    = 0,
            @trancountSave      = @@TRANCOUNT,
            @sid                = NULL

    EXECUTE AS CALLER
       SELECT @is_sysadmin       = IS_SRVROLEMEMBER('sysadmin'),
              @send_request_user = SUSER_SNAME(),
              @database_user_id  = USER_ID()
    REVERT

    --Check if SSB is enabled in this database
```

```
    --IF (ISNULL(DATABASEPROPERTYEX(DB_NAME(), N'IsBrokerEnabled'), 0) <> 1)
    --BEGIN
    --   RAISERROR(14650, 16, 1)
    --   RETURN 1
    --END

    --Report error if the mail queue has been stopped.
    --sysmail_stop_sp/sysmail_start_sp changes the receive status of the SSB queue
    --IF NOT EXISTS (SELECT * FROM sys.service_queues WHERE name = N'ExternalMailQueue' AND
is_receive_enabled = 1)
    --BEGIN
    --   RAISERROR(14641, 16, 1)
    --   RETURN 1
    --END

    -- Get the relevant profile_id
    --
    IF (@profile_name IS NULL)
    BEGIN
        -- Use the global or users default if profile name is not supplied
        SELECT TOP (1) @profile_id = pp.profile_id
        FROM dbo.sysmail_principalprofile as pp
        WHERE (pp.is_default = 1) AND
            (dbo.get_principal_id(pp.principal_sid) = @database_user_id OR pp.principal_sid = 0x00)
        ORDER BY dbo.get_principal_id(pp.principal_sid) DESC

        --Was a profile found
        --IF(@profile_id IS NULL)
        --BEGIN
        --    -- Try a profile lookup based on Windows Group membership, if any
        --    EXEC @rc = dbo.sp_validate_user @send_request_user, @sid OUTPUT
        --    IF (@rc = 0)
        --    BEGIN
        --        SELECT TOP (1) @profile_id = pp.profile_id
        --        FROM dbo.sysmail_principalprofile as pp
        --        WHERE (pp.is_default = 1) AND
        --              (pp.principal_sid = @sid)
        --        ORDER BY dbo.get_principal_id(pp.principal_sid) DESC
        --    END

            IF(@profile_id IS NULL)
            BEGIN
                RAISERROR(14636, 16, 1)
                RETURN 1
            END
        --END
    END
    ELSE
    BEGIN
        --Get primary account if profile name is supplied
        EXEC @rc = dbo.sysmail_verify_profile_sp @profile_id = NULL,
                        @profile_name = @profile_name,
                        @allow_both_nulls = 0,
                        @allow_id_name_mismatch = 0,
                        @profileid = @profile_id OUTPUT
        IF (@rc <> 0)
            RETURN @rc

        --Make sure this user has access to the specified profile.
        --sysadmins can send on any profiles
        IF ( @is_sysadmin <> 1)
        BEGIN
            --Not a sysadmin so check users access to profile
            IF NOT EXISTS(SELECT *
                        FROM dbo.sysmail_principalprofile
                        WHERE ((profile_id = @profile_id) AND
                                (dbo.get_principal_id(principal_sid) = @database_user_id OR principal_sid
```

```
= 0x00)))
            BEGIN
                --EXEC dbo.sp_validate_user @send_request_user, @sid OUTPUT
                --IF(@sid IS NULL)
                --BEGIN
                    RAISERROR(14607, -1, -1, 'profile')
                    RETURN 1
                --END
            END
        END
    END

    --Attach results must be specified
    IF @attach_query_result_as_file IS NULL
    BEGIN
       RAISERROR(14618, 16, 1, 'attach_query_result_as_file')
       RETURN 2
    END

    --No output must be specified
    IF @exclude_query_output IS NULL
    BEGIN
       RAISERROR(14618, 16, 1, 'exclude_query_output')
       RETURN 3
    END

    --No header must be specified
    IF @query_result_header IS NULL
    BEGIN
       RAISERROR(14618, 16, 1, 'query_result_header')
       RETURN 4
    END

    -- Check if query_result_separator is specifed
    IF @query_result_separator IS NULL OR DATALENGTH(@query_result_separator) = 0
    BEGIN
       RAISERROR(14618, 16, 1, 'query_result_separator')
       RETURN 5
    END

    --Echo error must be specified
    IF @append_query_error IS NULL
    BEGIN
       RAISERROR(14618, 16, 1, 'append_query_error')
       RETURN 6
    END

    --@body_format can be TEXT (default) or HTML
    IF (@body_format IS NULL)
    BEGIN
       SET @body_format = 'TEXT'
    END
    ELSE
    BEGIN
       SET @body_format = UPPER(@body_format)

       IF @body_format NOT IN ('TEXT', 'HTML')
       BEGIN
          RAISERROR(14626, 16, 1, @body_format)
          RETURN 13
       END
    END

    --Importance must be specified
    IF @importance IS NULL
    BEGIN
       RAISERROR(14618, 16, 1, 'importance')
```

```
   RETURN 15
END

SET @importance = UPPER(@importance)

--Importance must be one of the predefined values
IF @importance NOT IN ('LOW', 'NORMAL', 'HIGH')
BEGIN
   RAISERROR(14622, 16, 1, @importance)
   RETURN 16
END

--Sensitivity must be specified
IF @sensitivity IS NULL
BEGIN
   RAISERROR(14618, 16, 1, 'sensitivity')
   RETURN 17
END

SET @sensitivity = UPPER(@sensitivity)

--Sensitivity must be one of predefined values
IF @sensitivity NOT IN ('NORMAL', 'PERSONAL', 'PRIVATE', 'CONFIDENTIAL')
BEGIN
   RAISERROR(14623, 16, 1, @sensitivity)
   RETURN 18
END

--Message body cannot be null. Atleast one of message, subject, query,
--attachments must be specified.
IF( (@body IS NULL AND @query IS NULL AND @file_attachments IS NULL AND @subject IS NULL)
   OR
( (LEN(@body) IS NULL OR LEN(@body) <= 0)
   AND (LEN(@query) IS NULL  OR  LEN(@query) <= 0)
   AND (LEN(@file_attachments) IS NULL OR LEN(@file_attachments) <= 0)
   AND (LEN(@subject) IS NULL OR LEN(@subject) <= 0)
)
)
BEGIN
   RAISERROR(14624, 16, 1, '@body, @query, @file_attachments, @subject')
   RETURN 19
END
ELSE
   IF @subject IS NULL OR LEN(@subject) <= 0
      SET @subject='SQL Server Message'

--Recipients cannot be empty. Atleast one of the To, Cc, Bcc must be specified
IF ( (@recipients IS NULL AND @copy_recipients IS NULL AND
   @blind_copy_recipients IS NULL
    )
   OR
    ( (LEN(@recipients) IS NULL OR LEN(@recipients) <= 0)
   AND (LEN(@copy_recipients) IS NULL OR LEN(@copy_recipients) <= 0)
   AND (LEN(@blind_copy_recipients) IS NULL OR LEN(@blind_copy_recipients) <= 0)
    )
)
BEGIN
   RAISERROR(14624, 16, 1, '@recipients, @copy_recipients, @blind_copy_recipients')
   RETURN 20
END

--If query is not specified, attach results and no header cannot be true.
IF ( (@query IS NULL OR LEN(@query) <= 0) AND @attach_query_result_as_file = 1)
BEGIN
   RAISERROR(14625, 16, 1)
   RETURN 21
END
```

```
    --
    -- Execute Query if query is specified
    --IF ((@query IS NOT NULL) AND (LEN(@query) > 0))
    --BEGIN
    --    EXECUTE AS CALLER
    --    EXEC @rc = sp_RunMailQuery
    --                @query                     = @query,
    --            @attach_results            = @attach_query_result_as_file,
    --                @query_attachment_filename = @query_attachment_filename,
    --            @no_output                 = @exclude_query_output,
    --            @query_result_header       = @query_result_header,
    --            @separator                 = @query_result_separator,
    --            @echo_error                = @append_query_error,
    --            @dbuse                     = @execute_query_database,
    --            @width                     = @query_result_width,
    --             @temp_table_uid            = @temp_table_uid,
    --        @query_no_truncate         = @query_no_truncate,
    --        @query_result_no_padding           = @query_result_no_padding
    --  -- This error indicates that query results size was over the configured MaxFileSize.
    --  -- Note, an error has already beed raised in this case
    --  IF(@rc = 101)
    --     GOTO ErrorHandler;
    --     REVERT

    --     -- Always check the transfer tables for data. They may also contain error messages
    --     -- Only one of the tables receives data in the call to sp_RunMailQuery
    --     IF(@attach_query_result_as_file = 1)
    --     BEGIN
    --        IF EXISTS(SELECT * FROM sysmail_attachments_transfer WHERE uid = @temp_table_uid)
    --        SET @AttachmentsExist = 1
    --     END
    --     ELSE
    --     BEGIN
    --         IF EXISTS(SELECT * FROM sysmail_query_transfer WHERE uid = @temp_table_uid AND uid IS NOT
NULL)
    --        SET @QueryResultsExist = 1
    --     END

    --     -- Exit if there was an error and caller doesn't want the error appended to the mail
    --     IF (@rc <> 0 AND @append_query_error = 0)
    --     BEGIN
    --        --Error msg with be in either the attachment table or the query table
    --        --depending on the setting of @attach_query_result_as_file
    --        IF(@attach_query_result_as_file = 1)
    --        BEGIN
    --           --Copy query results from the attachments table to mail body
    --           SELECT @RetErrorMsg = CONVERT(NVARCHAR(4000), attachment)
    --           FROM sysmail_attachments_transfer
    --           WHERE uid = @temp_table_uid
    --        END
    --        ELSE
    --        BEGIN
    --           --Copy query results from the query table to mail body
    --           SELECT @RetErrorMsg = text_data
    --           FROM sysmail_query_transfer
    --           WHERE uid = @temp_table_uid
    --        END

    --        GOTO ErrorHandler;
    --     END
    --     SET @AttachmentsExist = @attach_query_result_as_file
    --END
    --ELSE
    --BEGIN
    --    --If query is not specified, attach results cannot be true.
    --    IF (@attach_query_result_as_file = 1)
```

```
--    BEGIN
--       RAISERROR(14625, 16, 1)
--       RETURN 21
--    END
--END

--Get the prohibited extensions for attachments from sysmailconfig.
--IF ((@file_attachments IS NOT NULL) AND (LEN(@file_attachments) > 0))
--BEGIN
--    EXECUTE AS CALLER
--    EXEC @rc = sp_GetAttachmentData
--                    @attachments = @file_attachments,
--                    @temp_table_uid = @temp_table_uid,
--                    @exclude_query_output = @exclude_query_output
--    REVERT
--    IF (@rc <> 0)
--        GOTO ErrorHandler;

--    IF EXISTS(SELECT * FROM sysmail_attachments_transfer WHERE uid = @temp_table_uid)
--        SET @AttachmentsExist = 1
--END

-- Start a transaction if not already in one.
-- Note: For rest of proc use GOTO ErrorHandler for falures
if (@trancountSave = 0)
   BEGIN TRAN @procName

SET @tranStartedBool = 1

-- Store complete mail message for history/status purposes
INSERT sysmail_mailitems
(
   profile_id,
   recipients,
   copy_recipients,
   blind_copy_recipients,
   subject,
   body,
   body_format,
   importance,
   sensitivity,
   file_attachments,
   attachment_encoding,
   query,
   execute_query_database,
   attach_query_result_as_file,
   query_result_header,
   query_result_width,
   query_result_separator,
   exclude_query_output,
   append_query_error,
   send_request_user,
   from_address,
   reply_to
)
VALUES
(
   @profile_id,
   @recipients,
   @copy_recipients,
   @blind_copy_recipients,
   @subject,
   @body,
   @body_format,
   @importance,
   @sensitivity,
   @file_attachments,
```

```
       'MIME',
       @query,
       @execute_query_database,
       @attach_query_result_as_file,
       @query_result_header,
       @query_result_width,
       @query_result_separator,
       @exclude_query_output,
       @append_query_error,
       @send_request_user,
       @from_address,
       @reply_to
    )

    SELECT @rc          = @@ERROR,
           @mailitem_id = SCOPE_IDENTITY()

    IF(@rc <> 0)
        GOTO ErrorHandler;

    --Copy query into the message body
   -- IF(@QueryResultsExist = 1)
   -- BEGIN
   --    -- if the body is null initialize it
   --      UPDATE sysmail_mailitems
   --      SET body = N''
   --      WHERE mailitem_id = @mailitem_id
   --      AND body is null

   --      --Add CR, a \r followed by \n, which is 0xd and then 0xa
   --      SET @CR_str = CHAR(13) + CHAR(10)
   --      UPDATE sysmail_mailitems
   --      SET body.WRITE(@CR_str, NULL, NULL)
   --      WHERE mailitem_id = @mailitem_id

   ----Copy query results to mail body
   --      UPDATE sysmail_mailitems
   --      SET body.WRITE( (SELECT text_data from sysmail_query_transfer WHERE uid = @temp_table_uid),
NULL, NULL )
   --      WHERE mailitem_id = @mailitem_id
   -- END

    --Copy into the attachments table
    --IF(@AttachmentsExist = 1)
    --BEGIN
    --    --Copy temp attachments to sysmail_attachments
    --    INSERT INTO sysmail_attachments(mailitem_id, filename, filesize, attachment)
    --    SELECT @mailitem_id, filename, filesize, attachment
    --    FROM sysmail_attachments_transfer
    --    WHERE uid = @temp_table_uid
    --END

    -- Create the primary SSB xml maessage
    --SET @sendmailxml = '<requests:SendMail xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.microsoft.com/databasemail/requests RequestTypes.xsd"
xmlns:requests="http://schemas.microsoft.com/databasemail/requests"><MailItemId>'
    --                    + CONVERT(NVARCHAR(20), @mailitem_id) + N'</MailItemId></requests:SendMail>'

    -- Send the send request on queue.
    --EXEC @rc = sp_SendMailQueues @sendmailxml
    --IF @rc <> 0
    --BEGIN
    --   RAISERROR(14627, 16, 1, @rc, 'send mail')
    --   GOTO ErrorHandler;
    --END

    -- Print success message if required
```

```
    IF (@exclude_query_output = 0)
    BEGIN
       SET @localmessage = FORMATMESSAGE(14635)
       PRINT @localmessage
    END

    --
    -- See if the transaction needs to be commited
    --
    IF (@trancountSave = 0 and @tranStartedBool = 1)
       COMMIT TRAN @procName

    -- All done OK
    goto ExitProc;

    -----------------
    -- Error Handler
    -----------------
ErrorHandler:
    IF (@tranStartedBool = 1)
       ROLLBACK TRAN @procName

    ------------------
    -- Exit Procedure
    ------------------
ExitProc:

    --Always delete query and attactment transfer records.
   --Note: Query results can also be returned in the sysmail_attachments_transfer table
    --DELETE sysmail_attachments_transfer WHERE uid = @temp_table_uid
    --DELETE sysmail_query_transfer WHERE uid = @temp_table_uid

   --Raise an error it the query execution fails
   -- This will only be the case when @append_query_error is set to 0 (false)
   IF( (@RetErrorMsg IS NOT NULL) AND (@exclude_query_output=0) )
   BEGIN
      RAISERROR(14661, -1, -1, @RetErrorMsg)
   END

    RETURN (@rc)
END
GO

-- ================================================
-- TEST DI INVIO
-- ================================================

EXECUTE dbo.sysmail_add_account_sp
    @account_name = 'Account DBMail',
    @description = 'Primo account Database Mail.',
    @email_address = 'vittorio@azure.com',
    @replyto_address = 'polizzi@azure.com',
    @display_name = 'Automated Mailer',
    @mailserver_name = 'smtp.azure.com',
    @username = 'vittorio.polizzi',
    @password = '53cr3t.pa$$w0rd';

-- Creazione di un profilo Database Mail
EXECUTE dbo.sysmail_add_profile_sp
    @profile_name = 'DBMail Profile',
    @description = 'Profilo Database Mail.' ;

-- Aggiunta dell'account nel profilo
EXECUTE dbo.sysmail_add_profileaccount_sp
    @profile_name = 'DBMail Profile',
    @account_name = 'Account DBMail',
    @sequence_number =1 ;
```

```
-- Concessione agli utenti dell'accesso al profilo
EXECUTE dbo.sysmail_add_principalprofile_sp
    @profile_name = 'DBMail Profile',
    @principal_name = 'public',
    @is_default = 1 ;

EXEC sp_send_dbmail @profile_name=NULL,
@recipients='test@contoso.com',
@subject='Messaggio di prova',
@body='Questo è il testo del messaggio.',
@file_attachments='attachment_file.txt'

SELECT     sysmail_mailitems.recipients, sysmail_mailitems.copy_recipients,
sysmail_mailitems.blind_copy_recipients, sysmail_mailitems.subject, sysmail_mailitems.body,
                      sysmail_mailitems.body_format, sysmail_mailitems.importance,
sysmail_mailitems.sensitivity, sysmail_mailitems.file_attachments, sysmail_mailitems.sent_status,
                      sysmail_account.email_address, sysmail_account.display_name,
sysmail_account.replyto_address, sysmail_server.servername, sysmail_server.port,
                      sysmail_server.username, sysmail_profileaccount.sequence_number,
sysmail_account_credential.cyphertext AS [Password]
FROM         sysmail_profileaccount INNER JOIN
                      sysmail_account ON sysmail_profileaccount.account_id = sysmail_account.account_id
INNER JOIN
                      sysmail_mailitems INNER JOIN
                      sysmail_profile ON sysmail_mailitems.profile_id = sysmail_profile.profile_id ON
sysmail_profileaccount.profile_id = sysmail_profile.profile_id INNER JOIN
                      sysmail_server ON sysmail_account.account_id = sysmail_server.account_id LEFT OUTER
JOIN
                      sysmail_account_credential ON sysmail_server.credential_id =
sysmail_account_credential.credential_id
WHERE     (sysmail_mailitems.sent_status = 0)
ORDER BY sysmail_profileaccount.sequence_number
```