Use and manage Database Mail

SQL Server 2005 has a small mail system called Database Mail. As an improved feature in SQL Mail compared to earlier versions of SQL Server, Database Mail is a mail queue system. Email messages are stored in an internal queue

SQL Server 2005 has a small mail system called Database Mail. Database Mail is an improved feature in SQL Mail compared to previous versions of SQL Server. With this feature you can set up multiple accounts as well as profiles to support email problems. What is Database Mail?

Database Mail is a mail queue system. Email messages are stored in a queue within the database to be processed. When an email message is placed in the queue, an open process is activated to send messages in the queue to the appropriate mail server. When the email has been sent, an email message with the status of distribution will be sent back to SQL Server.

Activate Database Mail

Database Mail is not available when installed. From the SQL Server security model, some things are turned off by default, you need to enable Database Mail if you want to use it. You can use the Surface Area Configuration or T-SQL Server configuration tool below to enable Database Mail:

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Database Mail XPs', 1;
GO
RECONFIGURE
GO
```

Account settings

A Database Mail account shows how SQL Server 2005 must communicate with the SMTP server. The account specifies how the email is formatted and sent. An account will identify an SMTP server and the authentication method. The account used for Database Mail does not correspond to the login account of SQL Server.

When setting up an account, you need to specify enough information so that SQL Server 2005 can communicate with the SMTP server and verify it if necessary. Refer to the data sources online for a complete list of options for defining an account. You can set up an account using the Database Mail Configuration Wizard, which can be found under 'Database Mail' in the 'Management' folder within SQL Server Management Studio, or by using the procedure. save (SP) 'sysmail_add_account_sp'. This is a code that uses the above SP method to create an

account that allows communication with an SMTP server that does not require authentication:

```
EXECUTE msdb.dbo.sysmail_add_account_sp @account_name = 'Database Administrati
```

This account is called 'Database Administration Account', and has an email address of *ProdServer01@domainname.com* . One of the advantages of using Database Mail is that this email address is not necessarily a valid email account in the mail system. In addition, you can combine a 'reply to' address with your database mail account. In my example above, I defined *Greg.Larsen@domainname.com* as *@replyto_address* . So when someone receives an email from any of my automated email notification processes they will be able to reply to it and the email can be sent directly to me.

With Database Mail, you can set up multiple email accounts if you want. In the following section, I will explain some of the reasons why you can set up multiple Database Mail accounts.

Set up profiles and link it to accounts

Before sending a Database Mail to the distinguished SMTP server in an account, you need to associate the account with a profile and allow access to the profile. Database Mail profile is used to improve mail security. There are two types of profiles: public and private. Public profile is always applied to anyone who has access to msdb database and is a member of DatabaseMailUserRole in msdb database. Meanwhile, private profiles can only be used for special users who are allowed to access the private profile. A mail profile can be combined with one or more other accounts. You can manage profiles using the Database Mail Configuration Wizard, or use the same T-SQL Server command as set up for a mail profile below:

```
- Create a Database Mail profile
EXECUTE msdb.dbo.sysmail_add_profile_sp
@profile_name = 'Database Administration Profile',
@description = 'Mail Profile for use by DBA processes';
```

Here I have created a profile named 'Database Administration Profile'. Having a profile created doesn't mean you can use this profile to send email. You still need to combine it with at least one Database Mail account and one user inside msdb data. To do that I can run the following two commands:

```
- Thêm tài kho?n vào danh sách
EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
@profile_name = 'Database Administration Profile',
@account_name = 'Database Administration Account',
@sequence_number = 1;
- Grant access to profile
EXECUTE msdb.dbo.sysmail_add_principalprofile_sp
@profile_name = 'Database Administration Profile',
@principal_name = 'ProdServer01',
```

The first EXECUTE statement will combine my profile with an account. The second statement will combine the profile with the 'ProdServer01' msdb user database. These users need to be allowed to be members of DatabaseMailUserRole before they can send mail. When I combine this profile with a special user, this profile is interpreted as a private profile. To create a public profile you need to combine a profile with a 'public' database role.

Send Database Mail

SQL Server provides SP 'sp_send_dbmail' to send mail. The following syntax is used to call this SP:

sp_send_dbmail [[@profile_name =] ' *profile_name* ']
[, [@recipients =] ' *recipients* [*; . n*] ']
[, [@copy_recipients =] ' *copy_recipient* [*; . n*] ']
[, [@blind_copy_recipients =] ' *blind_copy_recipient* [*; . n*] ']
[, [@subject =] ' *subject* ']
[, [@body =] ' *body* ']
[, [@body_format =] ' *body_format* ']
[, [@importance =] ' *importance* ']
[, [@sensitivity =] ' *sensitivity* ']
[, [@file_attachments =] ' *attachment* [*; . n*] ']
[, [@query =] ' *query* ']
[, [@execute_query_database =] ' *execute_query_database* ']
[, [@attach_query_result_as_file =] *attach_query_result_as_file*]
[, [@query_attachment_filename =] *query_attachment_filename*]
[, [@query_result_header =] *query_result_header*]
[, [@query_result_width =] *query_result_width*]
[, [@query_result_separator =] ' *query_result_separator* ']
[, [@exclude_query_output =] *exclude_query_output*]
[, [@append_query_error =] append *_query_error*]
[, [@query_no_truncate =] *query_no_truncate*]
[, [@mailitem_id =] *mailitem_id*] [OUTPUT]

As you can see, this SP supports a number of different parameters. Information on each of these parameters can be found in the online books.

Here is an example of how I will use the private profile created above to send a simple email message regardless of the automatic database re-indexing process.

```
EXEC msdb.dbo.sp_send_dbmail @profile_name = 'Database Administration Profile'
```

Database Mail handles all queues when running the above command, I will have a message that is sent saying that the mail is put into the 'Mail queued' queue. When you execute sp_send_dbmail, the mail is not sent immediately but instead it is stored in the mail queue inside the msdb database. This SP activates an extended mail process (DatabaseMail90.exe) to run. This executable reads the mail in the queue and sends it to the appropriate mail server.

Different cases in account and profile use

There are a number of different ways to use the advantages of multiple accounts and profiles that Database Mail allows you to do with them.

One of the most obvious advantages of the multi-account feature is that you can configure your Database Mail profile to allow failover when one of the SMTP servers has problems. When adding accounts to a Database Mail profile you can provide them with a sequence of sequence_number. When sending a new email message, Database Mail tries to send it using the lowest string number first. If the sending process fails, Database Mail

will use the next higher number sequence. Database Mail will constantly perform this task until the mail has been successfully sent or all accounts have been tried.

Another option in this use is to support the delivery of mail messages to different email addresses. If you have many applications that need to send email, each application has its own email address. This will help the e-mail recipient to distinguish what process was sent by viewing the email address.

In addition, if you use private profiles, you can combine these profiles with other security policies. That way will allow you to control the private profile, msdb users will be allowed to use a separate Database Mail profile.

Check Database Mail

SQL Server provides 6 different system windows in the msdb database for checking Database Mail information. These windows can be used to retrieve information in the msdb database that includes both the status of all Database Mail or only specific email messages. These windows are useful in distinguishing which mail has been processed as well as the reason why mail messages may not be delivered to the required mail server. They are the ideal tools that allow you to check and fix some Database Mail issues. For more details on these windows, you can find out more in online books.

sysmail_allitems - this window allows you to return a set of records, including one row for each email message processed by Database Mail.

sysmail_event_log - This window allows you to return a row for each Windows or SQL Server error message when Database Mail handles an email message.

sysmail_faileditems - This window returns a record for each email message that has a status or failure.

sysmail_mailattachments - This window contains one row for each sent attachment

sysmail_sentitems - This window contains a record for each email that has been sent successfully

sysmail_unsentitems - This window contains a record for each email currently in the queue to be sent or during the process being submitted.

Maintain notifications in MSDB Database

All email messages are stored in the msdb database, you must consider how to manage this information. Depending on the email memorization policies, you need to establish a periodic routine to clean up unnecessary email messages. SQL Server 2005 provides two different stored procedures to remove mail logs from the msdb database.

sysmail_delete_mailitems_sp - This SP regularly deletes the email messages contained in msdb in the Database Mail table.

sysmail_delete_log_sp - This SP deletes Database Mail log messages.

You can view books online for more details about SP.

T-SQL Server below will delete mail sent a month ago:

```
DECLARE @delete_date datetime
SET @delete_date = dateadd (MM, -1, getdate ())
EXECUTE msdb.dbo.sysmail_delete_mailitems_sp @ sent_before = @ delete_date
```

Conclude

Database Mail SQL Server 2005 is a major improvement in SQL Mail compared to previous versions of SQL Server. Database Mail provides a mail system with a lot of features to better protect and manage incoming mail using T-SQL. With Database Mail in SQL Server 2005, you don't need to build a solution for a solution using CDOSYS to send email. If you're looking to support email from T-SQL, consider Database Mail's features.