**Database Mail**

Database Mail is an enterprise solution for sending e-mail messages from the SQL Server Database Engine. Using Database Mail, your database applications can send e-mail messages to users. The messages can contain query results, and can also include files from any resource on your network.

Benefits of using Database Mail

Database Mail is designed for reliability, scalability, security, and supportability.

Reliability

* Database Mail uses the standard Simple Mail Transfer Protocol (SMTP) to send mail. You can use Database Mail without installing an Extended MAPI client on the computer that runs SQL Server.
* Process isolation. To minimize the impact on SQL Server, the component that delivers e-mail runs outside of SQL Server, in a separate process. SQL Server will continue to queue e-mail messages even if the external process stops or fails. The queued messages will be sent once the outside process or SMTP server comes online.
* Failover accounts. A Database Mail profile allows you to specify more than one SMTP server. Should an SMTP server be unavailable, mail can still be delivered to another SMTP server.
* Cluster support. Database Mail is cluster-aware and is fully supported on a cluster.

Scalability

* Background Delivery: Database Mail provides background, or asynchronous, delivery. When you call **sp\_send\_dbmail** to send a message, Database Mail adds a request to a Service Broker queue. The stored procedure returns immediately. The external e-mail component receives the request and delivers the e-mail.
* Multiple profiles: Database Mail allows you to create multiple profiles within a SQL Server instance. Optionally, you can choose the profile that Database Mail uses when you send a message.
* Multiple accounts: Each profile can contain multiple failover accounts. You can configure different profiles with different accounts to distribute e-mail across multiple e-mail servers.
* 64-bit compatibility: Database Mail is fully supported on 64-bit installations of SQL Server.

Security

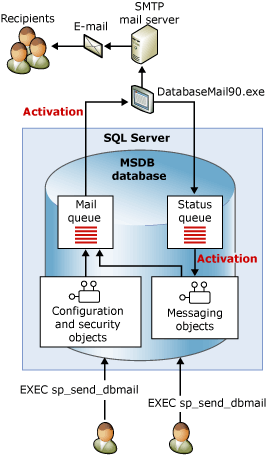
* Mail Security:To send Database Mail, you must be a member of the **DatabaseMailUserRole** database role in the **msdb** database.
* Profile security: Database Mail enforces security for mail profiles. You choose the **msdb** database users or groups that have access to a Database Mail profile. You can grant access to either specific users, or all users in **msdb**. A private profile restricts access to a specified list of users. A public profile is available to all users in a database.
* Attachment size governor: Database Mail enforces a configurable limit on the attachment file size. You can change this limit by using the [sysmail\_configure\_sp](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sysmail-configure-sp-transact-sql) stored procedure.
* Prohibited file extensions: Database Mail maintains a list of prohibited file extensions. Users cannot attach files with an extension that appears in the list. You can change this list by using sysmail\_configure\_sp.
* Database Mail runs under the SQL Server Engine service account. To attach a file from a folder to an email, the SQL Server engine account should have permissions to access the folder with the file.

Supportability

* Integrated configuration: Database Mail maintains the information for e-mail accounts within SQL Server Database Engine. There is no need to manage a mail profile in an external client application. Database Mail Configuration Wizard provides a convenient interface for configuring Database Mail. You can also create and maintain Database Mail configurations using Transact-SQL.
* Logging. Database Mail logs e-mail activity to SQL Server, the Microsoft Windows application event log, and to tables in the **msdb**database.
* Auditing: Database Mail keeps copies of messages and attachments sent in the **msdb** database. You can easily audit Database Mail usage and review the retained messages.
* Support for HTML: Database Mail allows you to send e-mail formatted as HTML.

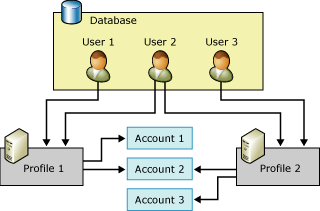
Database Mail Architecture

Database Mail is designed on a queued architecture that uses service broker technologies. When users execute **sp\_send\_dbmail**, the stored procedure inserts an item into the mail queue and creates a record that contains the e-mail message. Inserting the new entry in the mail queue starts the external Database Mail process (DatabaseMail.exe).



Database Mail Configuration Object Relationship

The illustration shows two profiles, three accounts, and three users. User 1 has access to Profile 1, which uses Account 1 and Account 2. User 3 has access to Profile 2, which uses Account 2 and Account 3. User 2 has access to both Profile 1 and Profile 2.



Database Mail Account

A Database Mail account contains the information that Microsoft SQL Server uses to send e-mail messages to an SMTP server. Each account contains information for one e-mail server.

A Database Mail supports three methods of authentication to communicate with an SMTP server:

* Windows Authentication: Database Mail uses the credentials of the SQL Server Database Engine Windows service account for authentication on the SMTP server.
* Basic Authentication: Database Mail uses the username and password specified to authenticate on the SMTP server.
* Anonymous Authentication: The SMTP server does not require any authentication. Database Mail will not use any credentials to authenticate on the SMTP server.

Account information is stored in the **msdb** database. Each account consists of the following information:

* The name of the account.
* A description of the account.
* The e-mail address of the account.
* The display name for the account.
* The e-mail address to use as the reply-to information for the account.
* The name of the e-mail server.
* The type of the e-mail server. For Microsoft SQL Server, this is always Simple Mail Transfer Protocol(SMTP).
* The port number of the e-mail server.
* A bit column indicating whether the connection to the SMTP mail server is made using Secure Sockets Layer (SSL).
* A bit column indicating whether the connection to the SMTP server is made using the credentials configured for the SQL Server Database Engine.
* The user name to use for authentication to the e-mail server, if the e-mail server requires authentication.
* The password to use for authentication to the e-mail server, if the e-mail server requires authentication.

The Database Mail Configuration Wizard provides a convenient way to create and manage accounts. You can also use the configuration stored procedures in **msdb** to create and manage accounts.

Database Mail Profile

A Database Mail profile is an ordered collection of related Database Mail accounts. Applications that send e-mail using Database Mail specify profiles, instead of using accounts directly. Separating information about the individual e-mail servers from the objects that the application uses improves flexibility and reliability: profiles provide automatic failover, so that if one e-mail server is unresponsive, Database Mail can automatically send mail to another e-mail server. Database administrators can add, remove, or reconfigure accounts without requiring changes to application code or job steps.

-- Create a Database Mail account

EXECUTE msdb.dbo.sysmail\_add\_account\_sp

@account\_name = 'AdventureWorks Administrator',

@description = 'Mail account for administrative e-mail.',

@email\_address = 'dba@Adventure-Works.com',

@replyto\_address = 'danw@Adventure-Works.com',

@display\_name = 'AdventureWorks Automated Mailer',

@mailserver\_name = 'smtp.Adventure-Works.com' ;

-- Create a Database Mail profile

EXECUTE msdb.dbo.sysmail\_add\_profile\_sp

@profile\_name = 'AdventureWorks Administrator Profile',

@description = 'Profile used for administrative mail.' ;

-- Add the account to the profile

EXECUTE msdb.dbo.sysmail\_add\_profileaccount\_sp

@profile\_name = 'AdventureWorks Administrator Profile',

@account\_name = 'AdventureWorks Administrator',

@sequence\_number =1 ;

-- Grant access to the profile to the DBMailUsers role

EXECUTE msdb.dbo.sysmail\_add\_principalprofile\_sp

@profile\_name = 'AdventureWorks Administrator Profile',

@principal\_name = 'ApplicationUser',

@is\_default = 1 ;