Entity Framework Migrations are handled from the package manager console in Visual Studio. The usage is shown in various tutorials, but I haven’t found a complete list of the commands available and their usage, so I created my own. There are four available main commands.

* [Enable-Migrations](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Enable-Migrations): Enables Code First Migrations in a project.
* [Add-Migration](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Add-Migration): Scaffolds a migration script for any pending model changes.
* [Update-Database](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Update-Database): Applies any pending migrations to the database.
* [Get-Migrations](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Get-Migrations): Displays the migrations that have been applied to the target database.

*This post was updated 2014-07-02 with Entity Framework 6.1.1*  
  
There are also three extra commands that are used by NuGet packages that install Entity Framework providers. These commands are not usually used as part of normal application development.

* [Add-EFProvider](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Add-EFProvider): Adds or updates an Entity Framework provider entry in the project config file.
* [Add-EFDefaultConnectionFactory](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Add-EFDefaultConnectionFactory): Adds or updates an Entity Framework default connection factory in the project config file.
* [Initialize-EFConfiguration](https://coding.abel.nu/2012/03/ef-migrations-command-reference/#Initialize-EFConfiguration): Initializes the Entity Framework section in the project config file and sets defaults.

The information here is the output of running get-help command-name -detailed for each of the commands in the package manager console (running EF  6.1.1). I’ve also added some own comments where I think some information is missing. My own comments are placed under the *Additional Information* heading.

*Please note that all commands should be entered on the same line. I’ve added line breaks to avoid horizontal scrollbars.*

Enable-Migrations

Enables Code First Migrations in a project.

Syntax

|  |
| --- |
| Enable-Migrations [-ContextTypeName <String>] [-EnableAutomaticMigrations]  [-MigrationsDirectory <String>] [-ProjectName <String>] [-StartUpProjectName <String>]  [-ContextProjectName <String>] [-ConnectionStringName <String>] [-Force]  [-ContextAssemblyName <String>] [-AppDomainBaseDirectory <String>] [<CommonParameters>]    Enable-Migrations [-ContextTypeName <String>] [-EnableAutomaticMigrations]  [-MigrationsDirectory <String>] [-ProjectName <String>] [-StartUpProjectName <String>]  [-ContextProjectName <String>] -ConnectionString <String>  -ConnectionProviderName <String> [-Force] [-ContextAssemblyName <String>]  [-AppDomainBaseDirectory <String>] [<CommonParameters>] |

Description

Enables Migrations by scaffolding a migrations configuration class in the project. If the target database was created by an initializer, an initial migration will be created (unless automatic migrations are enabled via the EnableAutomaticMigrations parameter).

Parameters

-ContextTypeName <String>

Specifies the context to use. If omitted, migrations will attempt to locate a single context type in the target project.

-EnableAutomaticMigrations [<SwitchParameter>]

Specifies whether automatic migrations will be enabled in the scaffolded migrations configuration. If ommitted, automatic migrations will be disabled.

-MigrationsDirectory <String>

Specifies the name of the directory that will contain migrations code files. If omitted, the directory will be named “Migrations”.

-ProjectName <String>

Specifies the project that the scaffolded migrations configuration class will be added to. If omitted, the default project selected in package manager console is used.

-StartUpProjectName <String>

Specifies the configuration file to use for named connection strings. If omitted, the specified project’s configuration file is used.

-ContextProjectName <String>

Specifies the project which contains the DbContext class to use. If omitted, the context is assumed to be in the same project used for migrations.

-ConnectionStringName <String>

Specifies the name of a connection string to use from the application’s configuration file.

-ConnectionString <String>

Specifies the the connection string to use. If omitted, the context’s default connection will be used.

-ConnectionProviderName <String>

Specifies the provider invariant name of the connection string.

-Force [<SwitchParameter>]

Specifies that the migrations configuration be overwritten when running more than once for given project.

-ContextAssemblyName <String>

Specifies the name of the assembly which contains the DbContext class to use. Use this parameter instead of ContextProjectName when the context is contained in a referenced assembly rather than in a project of the solution.

-AppDomainBaseDirectory <String>

Specifies the directory to use for the app-domain that is used for running Migrations code such that the app-domain is able to find all required assemblies. This is an advanced option that should only be needed if the solution contains several projects such that the assemblies needed for the context and configuration are not all referenced from either the project containing the context or the project containing the migrations.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

Examples

|  |
| --- |
| -------------------------- EXAMPLE 1 --------------------------  C:\PS>Enable-Migrations    # Scaffold a migrations configuration in a project with only one context    -------------------------- EXAMPLE 2 --------------------------  C:\PS>Enable-Migrations -Auto    # Scaffold a migrations configuration with automatic migrations enabled for a project  # with only one context    -------------------------- EXAMPLE 3 --------------------------  C:\PS>Enable-Migrations -ContextTypeName MyContext -MigrationsDirectory DirectoryName    # Scaffold a migrations configuration for a project with multiple contexts  # This scaffolds a migrations configuration for MyContext and will put the configuration  # and subsequent configurations in a new directory called "DirectoryName" |

Remarks

To see the examples, type: get-help Enable-Migrations -examples.  
For more information, type: get-help Enable-Migrations -detailed.  
For technical information, type: get-help Enable-Migrations -full.

Additional Information

The flag for enabling automatic migrations is saved in the Migrations\Configuration.cs file, in the constructor. To later change the option, just change the assignment in the file.

|  |
| --- |
| **public** Configuration()  {  AutomaticMigrationsEnabled = **false**;  } |

Add-Migration

Scaffolds a migration script for any pending model changes.

Syntax

|  |
| --- |
| Add-Migration [-Name] <String> [-Force] [-ProjectName <String>] [-StartUpProjectName <String>]  [-ConfigurationTypeName <String>] [-ConnectionStringName <String>] [-IgnoreChanges]  [-AppDomainBaseDirectory <String>] [<CommonParameters>]    Add-Migration [-Name] <String> [-Force] [-ProjectName <String>] [-StartUpProjectName <String>]  [-ConfigurationTypeName <String>] -ConnectionString <String> -ConnectionProviderName <String>  [-IgnoreChanges] [-AppDomainBaseDirectory <String>] [<CommonParameters>] |

Description

Scaffolds a new migration script and adds it to the project.

Parameters

-Name <String>

Specifies the name of the custom script.

-Force [<SwitchParameter>]

Specifies that the migration user code be overwritten when re-scaffolding an existing migration.

-ProjectName <String>

Specifies the project that contains the migration configuration type to be used. If ommitted, the default project selected in package manager console is used.

-StartUpProjectName <String>

Specifies the configuration file to use for named connection strings. If omitted, the specified project’s configuration file is used.

-ConfigurationTypeName <String>

Specifies the migrations configuration to use. If omitted, migrations will attempt to locate a single migrations configuration type in the target project.

-ConnectionStringName <String>

Specifies the name of a connection string to use from the application’s configuration file.

-ConnectionString <String>

Specifies the the connection string to use. If omitted, the context’s default connection will be used.

-ConnectionProviderName <String>

Specifies the provider invariant name of the connection string.

-IgnoreChanges

Scaffolds an empty migration ignoring any pending changes detected in the current model. This can be used to create an initial, empty migration to enable Migrations for an existing database. N.B. Doing this assumes that the target database schema is compatible with the current model.

-AppDomainBaseDirectory <String>

Specifies the directory to use for the app-domain that is used for running Migrations code such that the app-domain is able to find all required assemblies. This is an advanced option that should only be needed if the solution contains several projects such that the assemblies needed for the context and configuration are not all referenced from either the project containing the context or the project containing the migrations.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

|  |
| --- |
| -------------------------- EXAMPLE 1 --------------------------  C:\PS>Add-Migration First    # Scaffold a new migration named "First"    -------------------------- EXAMPLE 2 --------------------------  C:\PS>Add-Migration First -IgnoreChanges    # Scaffold an empty migration ignoring any pending changes detected in the current model.  # This can be used to create an initial, empty migration to enable Migrations for an existing  # database. N.B. Doing this assumes that the target database schema is compatible with the  # current model. |

<CommonParameters>

Remarks

To see the examples, type: get-help Add-Migration -examples.  
For more information, type: get-help Add-Migration -detailed.  
For technical information, type: get-help Add-Migration -full.

Update-Database

Applies any pending migrations to the database.

Syntax

|  |
| --- |
| Update-Database [-SourceMigration <String>] [-TargetMigration <String>] [-Script] [-Force]  [-ProjectName <String>] [-StartUpProjectName <String>] [-ConfigurationTypeName <String>]  [-ConnectionStringName <String>] [-AppDomainBaseDirectory <String>] [<CommonParameters>]    Update-Database [-SourceMigration <String>] [-TargetMigration <String>] [-Script] [-Force]  [-ProjectName <String>] [-StartUpProjectName <String>] [-ConfigurationTypeName <String>]  -ConnectionString <String> -ConnectionProviderName <String>  [-AppDomainBaseDirectory <String>] [<CommonParameters>] |

Description

Updates the database to the current model by applying pending migrations.

Parameters

-SourceMigration <String>

Only valid with -Script. Specifies the name of a particular migration to use as the update’s starting point. If ommitted, the last applied migration in the database will be used.

-TargetMigration <String>

Specifies the name of a particular migration to update the database to. If ommitted, the current model will be used.

-Script

Generate a SQL script rather than executing the pending changes directly.

-Force

Specifies that data loss is acceptable during automatic migration of the database.

-ProjectName <String>

Specifies the project that contains the migration configuration type to be used. If ommitted, the default project selected in package manager console is used.

-StartUpProjectName <String>

Specifies the configuration file to use for named connection strings. If omitted, the specified project’s configuration file is used.

-ConfigurationTypeName <String>

Specifies the migrations configuration to use. If omitted, migrations will attempt to locate a single migrations configuration type in the target project.

-ConnectionStringName <String>

Specifies the name of a connection string to use from the application’s configuration file.

-ConnectionString <String>

Specifies the the connection string to use. If omitted, the context’s default connection will be used.

-ConnectionProviderName <String>

Specifies the provider invariant name of the connection string.

-AppDomainBaseDirectory <String>

Specifies the directory to use for the app-domain that is used for running Migrations code such that the app-domain is able to find all required assemblies. This is an advanced option that should only be needed if the solution contains several projects such that the assemblies needed for the context and configuration are not all referenced from either the project containing the context or the project containing the migrations.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

Examples

|  |
| --- |
| -------------------------- EXAMPLE 1 --------------------------  C:\PS>Update-Database    # Update the database to the latest migration    -------------------------- EXAMPLE 2 --------------------------  C:\PS>Update-Database -TargetMigration Second    # Update database to a migration named "Second"  # This will apply migrations if the target hasn't been applied or roll back migrations  # if it has    -------------------------- EXAMPLE 3 --------------------------  C:\PS>Update-Database -Script    # Generate a script to update the database from it's current state to the latest migration    -------------------------- EXAMPLE 4 --------------------------  C:\PS>Update-Database -Script -SourceMigration Second -TargetMigration First    # Generate a script to migrate the database from a specified start migration  # named "Second" to a specified target migration named "First"    -------------------------- EXAMPLE 5 --------------------------  C:\PS>Update-Database -Script -SourceMigration $InitialDatabase    # Generate a script that can upgrade a database currently at any version to the latest version.  # The generated script includes logic to check the \_\_MigrationsHistory table and only apply changes  # that haven't been previously applied.    -------------------------- EXAMPLE 6 --------------------------  C:\PS>Update-Database -TargetMigration $InitialDatabase    # Runs the Down method to roll-back any migrations that have been applied to the database |

Remarks

To see the examples, type: get-help Update-Database -examples.  
For more information, type: get-help Update-Database -detailed.  
For technical information, type: get-help Update-Database -full.

Additional Information

The command always runs any pending code-based migrations first. If the database is still incompatible with the model the additional changes required are applied as an separate automatic migration step if automatic migrations are enabled. If automatic migrations are disabled an error message is shown.

Get-Migrations

Displays the migrations that have been applied to the target database.

Syntax

|  |
| --- |
| Get-Migrations [-ProjectName <String>] [-StartUpProjectName <String>] [-ConfigurationTypeName <String>]  [-ConnectionStringName <String>] [-AppDomainBaseDirectory <String>] [<CommonParameters>]    Get-Migrations [-ProjectName <String>] [-StartUpProjectName <String>] [-ConfigurationTypeName <String>]  -ConnectionString <String> -ConnectionProviderName <String> [-AppDomainBaseDirectory <String>]  [<CommonParameters>] |

Description

Displays the migrations that have been applied to the target database.

Parameters

-ProjectName <String>

Specifies the project that contains the migration configuration type to be used. If ommitted, the default project selected in package manager console is used.

-StartUpProjectName <String>

Specifies the configuration file to use for named connection strings. If omitted, the specified project’s configuration file is used.

-ConfigurationTypeName <String>

Specifies the migrations configuration to use. If omitted, migrations will attempt to locate a single migrations configuration type in the target project.

-ConnectionStringName <String>

Specifies the name of a connection string to use from the application’s configuration file.

-ConnectionString <String>

Specifies the the connection string to use. If omitted, the context’s default connection will be used.

-ConnectionProviderName <String>

Specifies the provider invariant name of the connection string.

-AppDomainBaseDirectory <String>

Specifies the directory to use for the app-domain that is used for running Migrations code such that the app-domain is able to find all required assemblies. This is an advanced option that should only be needed if the solution contains several projects such that the assemblies needed for the context and configuration are not all referenced from either the project containing the context or the project containing the migrations.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

Remarks

To see the examples, type: get-help Get-Migrations -examples.  
For more information, type: get-help Get-Migrations -detailed.  
For technical information, type: get-help Get-Migrations -full.

Add-EFProvider

Adds or updates an Entity Framework provider entry in the project config file.

Syntax

Add-EFProvider [-Project] <Object> [-InvariantName] <String> [-TypeName] <String> [<CommonParameters>]

Description

Adds an entry into the ‘entityFramework’ section of the project config file for the specified provider invariant name and provider type. If an entry for the given invariant name already exists, then that entry is updated with the given type name, unless the given type name already matches, in which case no action is taken. The ‘entityFramework’ section is added if it does not exist. The config file is automatically saved if and only if a change was made.

This command is typically used only by Entity Framework provider NuGet packages and is run from the ‘install.ps1’ script.

Parameters

-Project <Object>

The Visual Studio project to update. When running in the NuGet install.ps1 script the ‘$project’ variable provided as part of that script should be used.

-InvariantName <String>

The provider invariant name that uniquely identifies this provider. For example, the Microsoft SQL Server provider is registered with the invariant name ‘System.Data.SqlClient’.

-TypeName <String>

The assembly-qualified type name of the provider-specific type that inherits from ‘System.Data.Entity.Core.Common.DbProviderServices’. For example, for the Microsoft SQL Server provider, this type is ‘System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer’.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

Remarks

To see the examples, type: get-help Add-EFProvider -examples.  
For more information, type: get-help Add-EFProvider -detailed.  
For technical information, type: get-help Add-EFProvider -full.

Add-EFDefaultConnectionFactory

Adds or updates an Entity Framework default connection factory in the project config file.

Syntax

|  |
| --- |
| Add-EFDefaultConnectionFactory [-Project] <Object> [-TypeName] <String>  [-ConstructorArguments <String[]>] [<CommonParameters>] |

Description

Adds an entry into the ‘entityFramework’ section of the project config file for the connection factory that Entity Framework will use by default when creating new connections by convention. Any existing entry will be overridden if it does not match. The ‘entityFramework’ section is added if it does not exist. The config file is automatically saved if and only if a change was made.

This command is typically used only by Entity Framework provider NuGet packages and is run from the ‘install.ps1’ script.

Parameters

-Project <Object>

The Visual Studio project to update. When running in the NuGet install.ps1 script the ‘$project’ variable provided as part of that script should be used.

-TypeName <String>

The assembly-qualified type name of the connection factory type that implements the ‘System.Data.Entity.Infrastructure.IDbConnectionFactory’ interface. For example, for the Microsoft SQL Server Express provider  
connection factory, this type is ‘System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework’.

-ConstructorArguments <String[]>

An optional array of strings that will be passed as arguments to the connection factory type constructor.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

Remarks

To see the examples, type: get-help Add-EFDefaultConnectionFactory -examples.  
For more information, type: get-help Add-EFDefaultConnectionFactory -detailed.  
For technical information, type: get-help Add-EFDefaultConnectionFactory -full.

Initialize-EFConfiguration

Initializes the Entity Framework section in the project config file and sets defaults.

Syntax

|  |
| --- |
| Initialize-EFConfiguration [-Project] <Object> [<CommonParameters>] |

Description

Creates the ‘entityFramework’ section of the project config file and sets the default connection factory to use SQL Express if it is running on the machine, or LocalDb otherwise. Note that installing a different provider may change the default connection factory. The config file is automatically saved if and only if a change was made.

In addition, any reference to ‘System.Data.Entity.dll’ in the project is removed.

This command is typically used only by Entity Framework provider NuGet packages and is run from the ‘install.ps1’ script.

Parameters

-Project <Object>

The Visual Studio project to update. When running in the NuGet install.ps1 script the ‘$project’ variable provided as part of that script should be used.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about\_CommonParameters](http://go.microsoft.com/fwlink/?LinkID=113216" \t "_blank).

Remarks

To see the examples, type: get-help Initialize-EFConfiguration -examples.  
For more information, type: get-help Initialize-EFConfiguration -detailed.  
For technical information, type: get-help Initialize-EFConfiguration -full.

Additional Information

The powershell commands are complex powershell functions, located in the tools\EntityFramework.psm1 file of the Entity Framework installation. The powershell code is mostly a wrapper around the System.Data.Entity.Migrations.MigrationsCommands found in the tools\EntityFramework\EntityFramework.PowerShell.dll file. First a MigrationsCommands object is instantiated with all configuration parameters. Then there is a public method on the MigrationsCommands object for each of the available commands.