

GURU99 Dokumentimi

Pjesa Dyte

Prof:Muzafer Shala

Ass:Laberion Zebica Student:Albion Burrniku

Kampusi:Ferizaj

WINDOWS FORMS APPLICATION

Deri më tani kemi parë se si të punojmë me C # për të krijuar aplikacione të bazuara në tastierë. Por, në një skenar të jetës reale ekipi normalisht përdor Visual Studio dhe C # për të krijuar ose Formularë Windows ose aplikacione të bazuara në Web.

* [Windows Forms Basics](https://www.guru99.com/c-sharp-windows-forms-application.html#1)
* [Hello World in Windows Forms](https://www.guru99.com/c-sharp-windows-forms-application.html#2)
* [Adding Controls to a form](https://www.guru99.com/c-sharp-windows-forms-application.html#3)
* [Event Handling for Controls](https://www.guru99.com/c-sharp-windows-forms-application.html#4)
* [Tree and PictureBox Control](https://www.guru99.com/c-sharp-windows-forms-application.html#5)

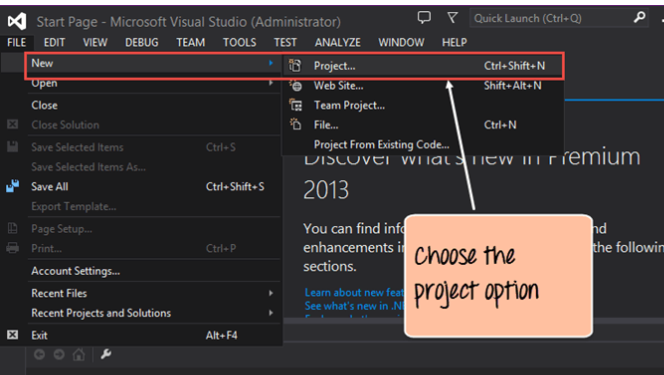
**Windows Forms Basics**

Një aplikacion i formave të Windows është ai që ekzekutohet në kompjuterin desktop. Një aplikacion i formave të Windows normalisht do të ketë një koleksion kontrollesh si etiketat, kutitë e teksteve, kutitë e listave, etj.

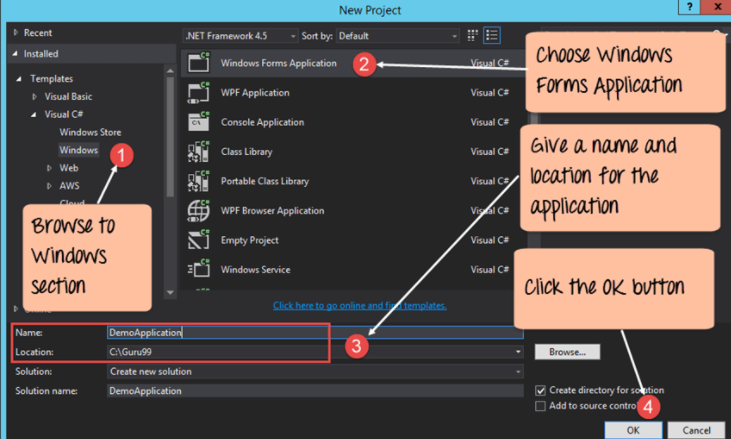
## C# Hello World

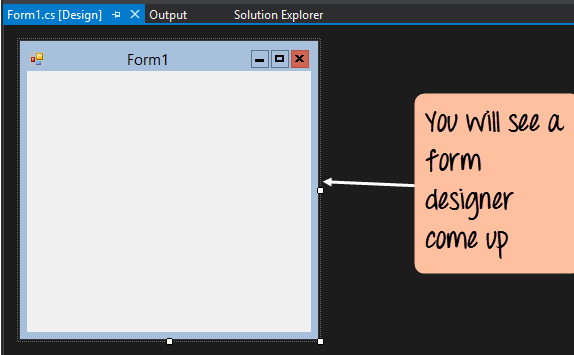
Tani le të shohim një shembull se si mund të implementojmë një aplikacion të thjeshtë 'hello world' në Visual Studio. Për këtë, do të duhet të zbatojmë hapat e përmendur më poshtë

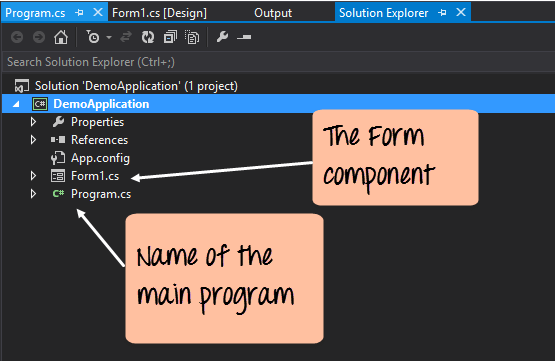
Hapi 1) Hapi i parë përfshin krijimin e një projekti të ri në Visual Studio. Pas nisjes së Visual Studio, duhet të zgjidhni opsionin e menusë New-> Project.



Hapi 2) Hapi tjetër është të zgjidhni llojin e projektit si një aplikacion Windows Forms. Këtu gjithashtu duhet të përmendim emrin dhe vendndodhjen e projektit tonë.







# **C# Database Connection: How to connect SQL Server**

Përdorimi i të dhënave nga një bazë e të dhënave është një nga aspektet e rëndësishme të çdo gjuhe programimi. Anshtë një domosdoshmëri absolute për çdo gjuhë programimi që të ketë aftësinë për të punuar me bazat e të dhënave. C # nuk është ndryshe.

Mund të punojë me lloje të ndryshme të bazave të të dhënave. Mund të punojë me bazat e të dhënave më të zakonshme si Oracle dhe Microsoft SQL Server.

Ai gjithashtu mund të punojë me forma të reja të bazave të të dhënave si MongoDB dhe MySQL.

* [Fundamentals of Database connectivity](https://www.guru99.com/c-sharp-access-database.html#1)
* [How to connect C# to Database](https://www.guru99.com/c-sharp-access-database.html#2)
* [Access data with the SqlDataReader](https://www.guru99.com/c-sharp-access-database.html#3)
* [C# Insert Into Database](https://www.guru99.com/c-sharp-access-database.html#4)
* [Updating Records](https://www.guru99.com/c-sharp-access-database.html#5)
* [Deleting Records](https://www.guru99.com/c-sharp-access-database.html#6)
* [Connecting Controls to Data](https://www.guru99.com/c-sharp-access-database.html#7)
* [C# DataGridView](https://www.guru99.com/c-sharp-access-database.html#8)

## Fundamentals of Database connectivity

C # dhe .Net mund të punojnë me shumicën e bazave të të dhënave, më e zakonshmja është Oracle dhe Microsoft SQL Server. Por, me çdo bazë të dhënash, logjika për të punuar me të gjithë është kryesisht e njëjtë.

Në shembujt tanë, ne do të shikojmë në funksionimin e Microsoft SQL Server si bazën tonë të të dhënave. Për qëllime të të mësuarit, mund të shkarkohet dhe të përdoret Microsoft SQL Server Express Edition, i cili është një softuer falas i bazës së të dhënave i siguruar nga Microsoft.

Në punën me bazat e të dhënave, në vijim janë konceptet që janë të përbashkëta për të gjitha bazat e të dhënave.

Lidhja - Për të punuar me të dhënat në një bazë të dhënash, hapi i parë i qartë është lidhja. Lidhja me një bazë të dhënash normalisht përbëhet nga parametrat e përmendur më poshtë.

Emri i bazës së të dhënave ose Burimi i të dhënave - Parametri i parë i rëndësishëm është emri i bazës së të dhënave me të cilën duhet të vendoset lidhja. Çdo lidhje mund të funksionojë vetëm me një bazë të dhënash në të njëjtën kohë.

Kredencialet - Aspekti tjetër i rëndësishëm është emri i përdoruesit dhe fjalëkalimi që duhet të përdoren për të vendosur një lidhje me bazën e të dhënave. Siguron që emri i përdoruesit dhe fjalëkalimi të kenë privilegjet e nevojshme për tu lidhur me bazën e të dhënave.

Parametrat opsionalë - Për secilin lloj të bazës së të dhënave, mund të specifikoni parametra opsionalë për të siguruar më shumë informacion se si .net duhet të trajtojë lidhjen me bazën e të dhënave. Për shembull, mund të specifikohet një parametër për sa kohë lidhja duhet të qëndrojë aktive. Nëse asnjë operacion nuk kryhet për një periudhë specifike kohore, atëherë parametri do të përcaktojë nëse lidhja duhet të mbyllet.

Përzgjedhja e të dhënave nga baza e të dhënave - Pasi të jetë vendosur lidhja, aspekti tjetër i rëndësishëm është marrja e të dhënave nga baza e të dhënave. C # mund të ekzekutojë komandën e zgjedhjes 'SQL' kundër bazës së të dhënave. Deklarata 'SQL' mund të përdoret për të marrë të dhëna nga një tabelë specifike në bazën e të dhënave.

Futja e të dhënave në bazën e të dhënave - C # mund të përdoret gjithashtu për të futur rekorde në bazën e të dhënave. Vlerat mund të specifikohen në C # për secilën rresht që duhet të futet në bazën e të dhënave.

Përditësimi i të dhënave në bazën e të dhënave - C # mund të përdoret gjithashtu për të azhurnuar të dhënat ekzistuese në bazën e të dhënave. Vlerat e reja mund të specifikohen në C # për secilën rresht që duhet të azhurnohet në bazën e të dhënave.

Fshirja e të dhënave nga një bazë të dhënash - C # mund të përdoret gjithashtu për të fshirë të dhënat në bazën e të dhënave. Zgjidhni komandat për të specifikuar se cilat rreshta duhet të fshihen mund të specifikohen në C #.

Ok, tani që kemi parë teorinë e secilit operacion, le të hyjmë në seksionet e mëtejshme për të parë se si mund të kryejmë operacione të bazës së të dhënave në C #

## SQL Command in c#

SqlCommand në C # lejojnë përdoruesin të kërkojë dhe të dërgojë komandat në bazën e të dhënave. Komanda SQL specifikohet nga objekti i lidhjes SQL. Përdoren dy metoda, metoda ExecuteReader për rezultatet e pyetjes dhe ExecuteNonQuery për komandat e futjes, azhurnimit dhe fshirjes. Theshtë metoda që është më e mira për komandat e ndryshme.

## How to connect C# to Database

Le të shohim tani kodin, i cili duhet të mbahet në vend për të krijuar një lidhje me një bazë të dhënash. Në shembullin tonë, ne do të lidhemi me një bazë të dhënash e cila ka emrin e Demodb. Kredencialet e përdorura për t'u lidhur me bazën e të dhënave janë dhënë më poshtë

* **Username – sa**
* **Password – demo123**

## KODI

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace DemoApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString;

SqlConnection cnn;

connetionString = @"Data Source=WIN-50GP30FGO75;Initial Catalog=Demodb;User ID=sa;Password=demol23";

cnn = new SqlConnection(connetionString);

cnn.Open();

MessageBox.Show("Connection Open !");

cnn.Close();

}

}

}

## Access data with the SqlDataReader

Për të shfaqur mënyrën se si mund të arrihen të dhënat duke përdorur C #, le të supozojmë se kemi artefaktet e mëposhtme në bazën tonë të të dhënave.

Një tryezë e quajtur demotb. Kjo tabelë do të përdoret për të ruajtur ID-në dhe emrat e udhëzimeve të ndryshme.

Tabela do të ketë 2 kolona, ​​njëra quhet "TutorialID" dhe tjetra quhet "TutorialName".

Për momentin, tabela do të ketë 2 rreshta siç tregohet më poshtë.

ID i udhëzuesit

Emri i udhëzuesit

1 C #

2 ASP.Net

Le të ndryshojmë kodin në formën tonë, në mënyrë që të mund të kërkojmë për këto të dhëna dhe të shfaqim informacionin përmes një Mesazhi. Vini re se i gjithë kodi i futur më poshtë është një vazhdim i kodit të shkruar për lidhjen e të dhënave në seksionin e mëparshëm.

# **C# File I/O Handling Operations**

C # ka një varg të gjerë veprimesh skedarësh. Këto operacione përfshijnë hapjen e një skedari, leximin ose shkrimin në një skedar. Mund të ketë raste kur dëshironi të punoni me skedarë direkt, në këtë rast do të përdorni operacionet e skedarëve të disponueshëm në C #. Disa nga veprimet themelore të skedarit janë përmendur më poshtë.

Leximi - Ky operacion është operacioni themelor i leximit, ku të dhënat lexohen nga një skedar.

Shkrimi - Ky operacion është operacioni themelor i shkrimit ku të dhënat shkruhen në një skedar. Si parazgjedhje, të gjitha përmbajtjet ekzistuese hiqen nga skedari dhe është shkruar përmbajtja e re.

Shtimi - Ky operacion përfshin gjithashtu shkrimin e informacionit në një skedar. I vetmi ndryshim është se të dhënat ekzistuese në një skedar nuk mbishkruhen. Të dhënat e reja që do të shkruhen shtohen në fund të skedarit.

KODI

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

if (File.Exists(path))

{

Console.WriteLine("File Exists");

}

Console.ReadKey();

}

}

}

## File.ReadAlllines

KODI

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

String[] lines;

lines = File.ReadAllLines(path);

Console.WriteLine(lines[0]);

Console.WriteLine(lines[1]);

Console.ReadKey();

}

}

}

## File.ReadAllText

KODI

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

String lines;

lines = File.ReadAllText(path);

Console.WriteLine(lines);

Console.ReadKey();

}

}

}

## File.Copy

KODI

using System;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

String copypath = @"D:\ExampleNew.txt";

File.Copy(path,copypath);

Console.ReadKey();

}

}

}

## File.Delete

KODI

using System;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

File.Delete(path);

Console.ReadKey();

}

}

}

|  |  |
| --- | --- |
| **File Method** | **Description** |
| File.Exists | File exists method is used to check if a particular file exists. |
| File.ReadAlllines | The method is used to read all the lines one by one in a file. |
| File.ReadAllText | This method is used to read all the lines in a file at once. |
| File.Copy | The method is used to make a copy of an existing file. |
| File.Delete | The method is used to delete an existing file. |

# **C# Stream Tutorial: StreamReader, StreamWriter**

Në operacionet e skedarit C #, normalisht rrjedhat përdoren për të lexuar dhe shkruar në skedarë. Një rrjedhë është një shtresë shtesë e krijuar midis një aplikacioni dhe një skedari. Transmetimi përdoret për të siguruar operacione të lehta leximi dhe shkrimi në skedar.

Rrymat zakonisht përdoren kur lexoni të dhëna nga skedarët e mëdhenj. Duke përdorur transmetime, të dhënat nga skedarët e mëdhenj ndahen në copa të vogla dhe dërgohen në lumë. Këto copa të dhënash mund të lexohen nga aplikacioni.

In this tutorial, you will learn-

* [Stream Reader](https://www.guru99.com/c-sharp-stream.html#1)
* [Stream Writer](https://www.guru99.com/c-sharp-stream.html#2)

## Stream Reader

Lexuesi i rrymës përdoret për të lexuar të dhëna nga një skedar duke përdorur rrjedha. Të dhënat nga skedari lexohen së pari në lumë. Pas kësaj, aplikacioni lexon të dhënat nga rryma.

KODI

using System;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

using (StreamReader sr = File.OpenText(path))

{

String s = "";

while ((s = sr.ReadLine()) != null)

{

Console.WriteLine(s);

}

}

Console.ReadKey();

}

}

}

## Stream Writer

Shkruesi i rrymës përdoret për të shkruar të dhëna në një skedar duke përdorur transmetime. Të dhënat nga aplikacioni shkruhen së pari në lumë. Pas kësaj rryma shkruan të dhënat në skedar. Le të shohim një shembull se si mund të përdorim rrjedhat për të shkruar të dhëna nga një skedar. Vendosni kodin më poshtë në skedarin program.cs.

KODI

using System;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

class Tutorial

{

static void Main(string[] args)

{

String path = @"D:\Example.txt";

using (StreamWriter sr = File.AppendText(path))

{

sr.WriteLine("Guru99 - ASP.Net");

sr.Close();

Console.WriteLine(File.ReadAllText(path));

}

Console.ReadKey();

}

}

}

# **C# Serialization & Deserialization**

Koncepti i Serializimit dhe deserializimit përdoret sa herë që të dhënat që kanë të bëjnë me objektet duhet të dërgohen nga një aplikacion në tjetrin. Serializimi përdoret për të eksportuar të dhëna të aplikacionit në një skedar. Aplikacioni i destinacionit pastaj përdor deserialization për të nxjerrë të dhënat nga aplikacioni për përdorim të mëtejshëm.

Serializimi është një koncept në të cilin objektet e klasës C # shkruhen ose serializohen në skedarë. Le të themi se keni pasur një klasë C # të quajtur Tutorial. Dhe klasa ka 2 veti të ID dhe emrin e Tutorials.

KODI

using System;

using System.IO;

using System.Linq;

using System.Runtime.Serialization;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

namespace DemoApplication

{

[Serializable]

class Tutorial

{

public int ID;

public String Name;

static void Main(string[] args)

{

Tutorial obj = new Tutorial();

obj.ID = 1;

obj.Name = ".Net";

IFormatter formatter = new BinaryFormatter();

Stream stream = new FileStream(@"E:\ExampleNew.txt",FileMode.Create,FileAccess.Write);

formatter.Serialize(stream, obj);

stream.Close();

stream = new FileStream(@"E:\ExampleNew.txt",FileMode.Open,FileAccess.Read);

Tutorial objnew = (Tutorial)formatter.Deserialize(stream);

Console.WriteLine(objnew.ID);

Console.WriteLine(objnew.Name);

Console.ReadKey();

}

}

}

# **10 Best C# IDE for Windows, Linux, Mac (2021 Update)**

Programimi në C # bëhet shumë i lehtë duke përdorur një ID # C. Ka shumë softuerë C # IDE të disponueshëm në treg që përmban një redaktues të kodit burimor, një korrigjues, mjete automatizimi të ndërtimit.

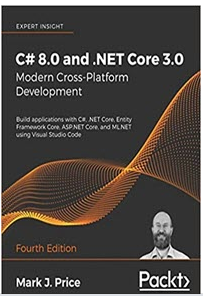
Më poshtë është një listë e zgjedhur me Top 9 C # IDE me karakteristikat e tyre të njohura dhe lidhjet e faqeve të internetit. Lista përmban softuer me burim të hapur (falas) dhe komercial (me pagesë).

## Best C# IDE

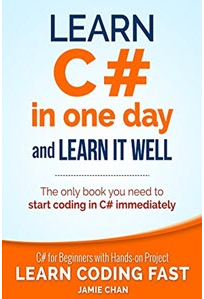
| **Name** | **Support platform** | **Link** |
| --- | --- | --- |
| [VS Code](https://code.visualstudio.com/) | MacOS, Windows, Linux. | [Learn More](https://code.visualstudio.com/) |
| [Kite](http://guru99.link/recommends-kite) | Windows | [Learn More](http://guru99.link/recommends-kite) |
| [SlickEdit](https://www.slickedit.com/) | Windows, Linux, Mac, and Raspberry Pi. | [Learn More](https://www.slickedit.com/) |
| [Eclipse aCute](https://projects.eclipse.org/projects/tools.acute) | Windows | [Learn](https://projects.eclipse.org/projects/tools.acute) |

# **14 BEST C# Books (2021 Update)**

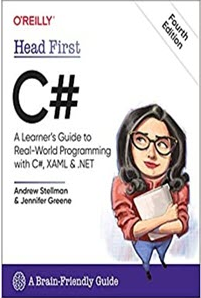
### 1) [C# 8.0 and .NET Core 3.0](https://geni.us/GmIE5p1)



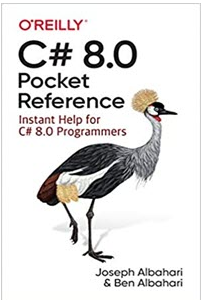
### 2) [C#: Learn C# in One Day and Learn It Well. C# for Beginners with Hands-on Project](https://geni.us/czxXAbV)



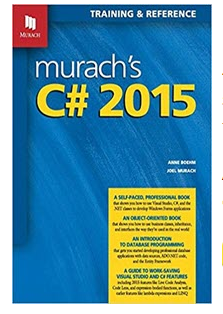
### 3) [Head First C#: A Learner's Guide to Real-World Programming with C#, XAML, and .NET](https://geni.us/DGszL)



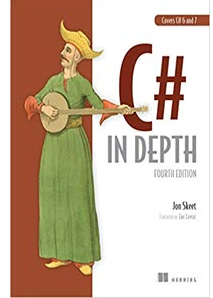
### 4) [C# 8.0 Pocket Reference: Instant Help for C# 8.0 Programmers](https://geni.us/XBcLua)



### 5) [Murach's C# 2015](https://geni.us/A1GAs)



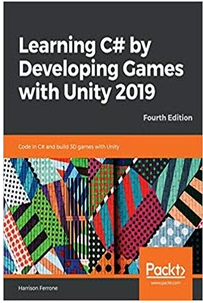
### 6) [C# in Depth](https://geni.us/KcQv4W)



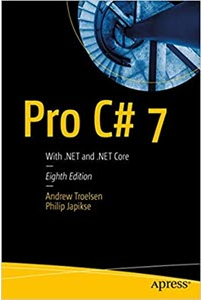
### 7) [The C# Player's Guide](https://geni.us/hfBE)



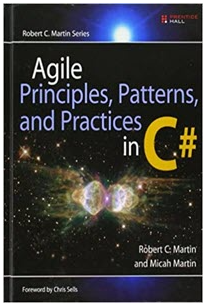
### 8) [Learning C# by Developing Games with Unity 2019](https://geni.us/bOLZ3)



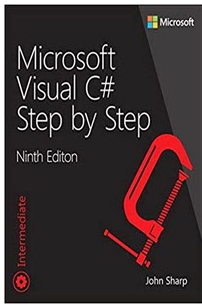
### 9) [Pro C# 7: With .NET and .NET Core](https://geni.us/WjRQYXd)



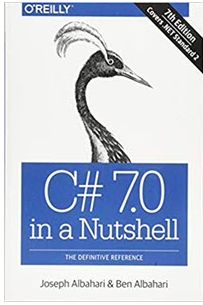
### 10) [Agile Principles, Patterns, and Practices in C#](https://geni.us/mkMU)



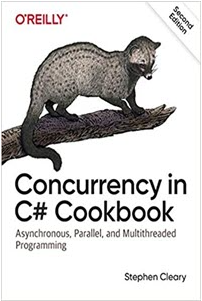
### 11) [Microsoft Visual C# Step by Step](https://geni.us/Oubixp)



### 12) [C# 7.0 in a Nutshell: The Definitive Reference](https://geni.us/6f64PZ)



### 13) [Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming](https://geni.us/zDIa2)



### 14) [Effective C# (Covers C# 6.0), (includes Content Update Program)](https://geni.us/iN1r5)

