Table of Contents

Namespace FCli

Classes

FallenCli

Fallen-cli facade class.

Class FallenCli

Fallen-cli facade class.

Inheritance

object
FallenCli

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli

Assembly: FCli.dll

Syntax

```
public class FallenCli
```

Remarks

Implemented as background service for the app hosting container.

Constructors

FallenCli(ICommandLineFormatter, IResources, ILogger<FallenCli>, IArgsParser, IToolExecutor, ICommandFactory)

Declaration

```
public FallenCli(ICommandLineFormatter formatter, IResources resources, ILogger<FallenCli> logger, IArgsParser
args, IToolExecutor toolExecutor, ICommandFactory commandFactory)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
ILogger<FallenCli>	logger	
IArgsParser	args	
IToolExecutor	toolExecutor	
ICommandFactory	commandFactory	

Methods

Execute(string[])

Executes main fcli logic.

Declaration

```
public void Execute(string[] cArgs)
```

Parameters

TYPE	NAME	DESCRIPTION
string[]	cArgs	

Namespace FCli.Exceptions

Classes

CommandNameException

Abstracts all situations when something is wrong with a command name.

CriticalException

Fallen-cli root exception that represents an unexpected behavior that cannot be programmatically processed.

FlagException

This exception has semantics of an error concerning command line flags.

IdentityException

This exception is raised when something goes bad with authentication or identity.

MailException

Represents errors concerning the mailing system.

ResourceNotLoadedException

Critical exception that get's thrown if resources are missing.

Class CommandNameException

Abstracts all situations when something is wrong with a command name.

Inheritance

object
Exception
SystemException
ArgumentException
CommandNameException

Implements

ISerializable

Inherited Members

ArgumentException.GetObjectData(SerializationInfo, StreamingContext)
ArgumentException.ThrowIfNullOrEmpty(string, string)
ArgumentException.Message
ArgumentException.ParamName
Exception.GetBaseException()
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Source
Exception.StackTrace
Exception.TargetSite
Exception.SerializeObjectState
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: FCli.Exceptions

Assembly: FCli.dll

Syntax

```
public class CommandNameException : ArgumentException, ISerializable
```

Constructors

CommandNameException()

Declaration

```
public CommandNameException()
```

CommandNameException(string?)

Declaration

```
public CommandNameException(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	

CommandNameException(string?, Exception?)

Declaration

```
public CommandNameException(string? message, Exception? innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	
Exception	innerException	

Implements

ISerializable

Class CriticalException

Fallen-cli root exception that represents an unexpected behavior that cannot be programmatically processed.

Inheritance

object
Exception
CriticalException
ResourceNotLoadedException

Implements

ISerializable

Inherited Members

Exception.GetBaseException()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Message
Exception.Source
Exception.StackTrace
Exception.TargetSite
Exception.SerializeObjectState
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: FCli.Exceptions

Assembly: FCli.dll

Syntax

```
public class CriticalException : Exception, ISerializable
```

Constructors

CriticalException()

Declaration

```
public CriticalException()
```

CriticalException(string?)

Declaration

```
public CriticalException(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	

CriticalException(string?, Exception?)

Declaration

```
public CriticalException(string? message, Exception? innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	
Exception	innerException	

Implements

ISerializable

Class FlagException

This exception has semantics of an error concerning command line flags.

Inheritance

object
Exception
SystemException
ArgumentException
FlagException

Implements

ISerializable

Inherited Members

ArgumentException.GetObjectData(SerializationInfo, StreamingContext)
ArgumentException.ThrowIfNullOrEmpty(string, string)
ArgumentException.Message
ArgumentException.ParamName
Exception.GetBaseException()
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Source
Exception.StackTrace
Exception.TargetSite
Exception.SerializeObjectState
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: FCli.Exceptions

Assembly: FCli.dll

Syntax

```
public class FlagException : ArgumentException, ISerializable
```

Constructors

FlagException()

Declaration

```
public FlagException()
```

FlagException(string?)

Declaration

```
public FlagException(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	

FlagException(string?, Exception?)

Declaration

```
public FlagException(string? message, Exception? innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	
Exception	innerException	

Implements

ISerializable

Class IdentityException

This exception is raised when something goes bad with authentication or identity.

Inheritance

object
Exception
IdentityException
MailException

Implements

ISerializable

Inherited Members

Exception.GetBaseException()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Message
Exception.Source
Exception.StackTrace
Exception.TargetSite
Exception.SerializeObjectState
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: FCli.Exceptions
Assembly: FCli.dll

Syntax

```
public class IdentityException : Exception, ISerializable
```

Constructors

IdentityException()

Declaration

```
public IdentityException()
```

IdentityException(string?)

Declaration

```
public IdentityException(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	

IdentityException(string?, Exception?)

Declaration

```
public IdentityException(string? message, Exception? innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	
Exception	innerException	

Implements

ISerializable

Class MailException

Represents errors concerning the mailing system.

Inheritance

object
Exception
IdentityException
MailException

Implements

ISerializable

Inherited Members

Exception.GetBaseException()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Message
Exception.Source
Exception.StackTrace
Exception.TargetSite
Exception.SerializeObjectState
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: FCli.Exceptions

Assembly: FCli.dll

Syntax

```
public class MailException : IdentityException, ISerializable
```

Constructors

MailException()

Declaration

```
public MailException()
```

MailException(string?)

Declaration

```
public MailException(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	

MailException(string?, Exception?)

Declaration

```
public MailException(string? message, Exception? innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	
Exception	innerException	

Implements

ISerializable

Class ResourceNotLoadedException

Critical exception that get's thrown if resources are missing.

Inheritance

object
Exception
CriticalException
ResourceNotLoadedException

Implements

ISerializable

Inherited Members

Exception.GetBaseException()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Message
Exception.Source
Exception.StackTrace
Exception.TargetSite
Exception.SerializeObjectState
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: FCli.Exceptions

Assembly: FCli.dll

Syntax

```
public class ResourceNotLoadedException : CriticalException, ISerializable
```

Constructors

ResourceNotLoadedException()

Declaration

```
public ResourceNotLoadedException()
```

ResourceNotLoadedException(string?)

Declaration

```
public ResourceNotLoadedException(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	

ResourceNotLoadedException(string?, Exception?)

Declaration

```
public ResourceNotLoadedException(string? message, Exception? innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	
Exception	innerException	

Implements

ISerializable

Namespace FCli.Models

Classes

Args

Wrapper class for the command line arguments.

Command

Abstraction for a recorded command.

Flag

Key-Value record that represents command line flag.

Group

Abstraction for a sequence of commands.

Class Args

Wrapper class for the command line arguments.

Inheritance

object
Args
ArgsParser

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models

Assembly: FCli.dll

Syntax

```
public class Args
```

Remarks

Parses arguments pretty reliably (not always).

Constructors

Args()

Protected default constructor.

Declaration

```
protected Args()
```

Args(string, string, List<Flag>)

Constructor with all command or tool parameters.

Declaration

```
protected Args(string selector, string arg, List<Flag> flags)
```

Parameters

TYPE	NAME	DESCRIPTION
string	selector	The name of the command or tool.
string	arg	Argument for the tool.
List<Flag>	flags	Given command line flags.

Remarks

It is privated cause `Args` are meant to be parsed.

Properties

Arg

Argument (usually path) for the tool.

Declaration

```
public string Arg { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

Remarks

Commands use `--options` flag.

Flags

List of all parsed flags.

Declaration

```
public List<Flag> Flags { get; }
```

Property Value

TYPE	DESCRIPTION
List<Flag>	

Remarks

`--` starter is dropped.

None

Points to empty Args object.

Declaration

```
public static Args None { get; }
```

Property Value

TYPE	DESCRIPTION
Args	

Selector

Command line identifier for the command or tool.

Declaration

```
public string Selector { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

Class Command

Abstraction for a recorded command.

Inheritance

object
Command
Group

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models

Assembly: FCli.dll

Syntax

```
[JsonDerivedType(typeof(Group), "group")]
[JsonDerivedType(typeof(Command), "command")]
public class Command
```

Remarks

All properties are mandatory except for Action, which can be added afterwards.

Properties

Action

Action contains the actual logic for command execution.

Declaration

```
[JsonIgnore]
public Action? Action { get; set; }
```

Property Value

TYPE	DESCRIPTION
Action	

Remarks

Nullable to support Factory pattern and deserialize more cleanly.

Name

Command line selector of the command.

Declaration

```
public string Name { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Options

Additional command line arguments if needed.

Declaration

```
public string Options { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Path

Path or URL to the resource.

Declaration

```
public string Path { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Shell

Specifies shell type if this is a shell command.

Declaration

```
public ShellType Shell { get; init; }
```

Property Value

TYPE	DESCRIPTION
ShellType	

Type

Describes the way of execution.

Declaration

```
public CommandType Type { get; init; }
```

Property Value

TYPE	DESCRIPTION
CommandType	

Methods

ToAlterRequest()

Transforms this command into an alteration request.

Declaration

```
public CommandAlterRequest ToAlterRequest()
```

Returns

TYPE	DESCRIPTION
CommandAlterRequest	AlterRequest for this command.

Class Flag

Key-Value record that represents command line flag.

Inheritance

object
Flag

Implements

IEquatable<Flag>

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models

Assembly: FCli.dll

Syntax

```
public record Flag : IEquatable<Flag>
```

Constructors

Flag(string, string)

Key-Value record that represents command line flag.

Declaration

```
public Flag(string Key, string Value)
```

Parameters

TYPE	NAME	DESCRIPTION
string	Key	Flag selector.
string	Value	Flag argument.

Properties

Key

Flag selector.

Declaration

```
public string Key { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Value

Flag argument.

Declaration

```
public string Value { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Implements

IEquatable<T>

Class Group

Abstraction for a sequence of commands.

Inheritance

object
Command
Group

Inherited Members

Command.Name
Command.Type
Command.Shell
Command.Path
Command.Options
Command.Action
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models

Assembly: FCli.dll

Syntax

```
[JsonSerializable(typeof(Group))]
public class Group : Command
```

Properties

Sequence

Command designators, stored in an execution sequence.

Declaration

```
public List<string> Sequence { get; init; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Methods

ToAlterRequest()

Transforms this group to an alteration request.

Declaration

```
public GroupAlterRequest ToAlterRequest()
```

Returns

TYPE	DESCRIPTION
GroupAlterRequest	AlterRequest for this group.

Namespace FCli.Models.Dtos

Classes

CommandAlterRequest

Necessary information for the command to be created or modified.

EmailHeaderResponse

Represents email summary.

EmailMessageResponse

Summary with body.

GroupAlterRequest

Contains necessary information for creation or overriding group commands.

IdentityChangeRequest

Used to request identity creation or override.

SendEmailRequest

Necessary info to send an email.

Class CommandAlterRequest

Necessary information for the command to be created or modified.

Inheritance

object
CommandAlterRequest

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Dtos

Assembly: FCli.dll

Syntax

```
public class CommandAlterRequest
```

Properties

Name

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Options

Declaration

```
public string Options { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Path

Declaration

```
public string Path { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Shell

Declaration

```
public ShellType Shell { get; set; }
```

Property Value

TYPE	DESCRIPTION
ShellType	

Type

Declaration

```
public CommandType Type { get; set; }
```

Property Value

TYPE	DESCRIPTION
CommandType	

Class EmailHeaderResponse

Represents email summary.

Inheritance

object
EmailHeaderResponse
EmailMessageResponse

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Dtos

Assembly: FCli.dll

Syntax

```
public class EmailHeaderResponse
```

Properties

Date

Declaration

```
public DateTime Date { get; set; }
```

Property Value

TYPE	DESCRIPTION
DateTime	

Index

Declaration

```
public int Index { get; }
```

Property Value

TYPE	DESCRIPTION
int	

IsRead

Declaration

```
public bool IsRead { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

SenderEmail

Declaration

```
public string SenderEmail { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

SenderName

Declaration

```
public string SenderName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Subject

Declaration

```
public string Subject { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Class EmailMessageResponse

Summary with body.

Inheritance

object
EmailHeaderResponse
EmailMessageResponse

Inherited Members

EmailHeaderResponse.Index
EmailHeaderResponse.SenderEmail
EmailHeaderResponse.SenderName
EmailHeaderResponse.Subject
EmailHeaderResponse.Date
EmailHeaderResponse.IsRead
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Dtos

Assembly: FCli.dll

Syntax

```
public class EmailMessageResponse : EmailHeaderResponse
```

Properties

Body

Declaration

```
public string Body { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Class GroupAlterRequest

Contains necessary information for creation or overriding group commands.

Inheritance

object

GroupAlterRequest

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: FCli.Models.Dtos

Assembly: FCli.dll

Syntax

```
public class GroupAlterRequest
```

Properties

Name

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Sequence

Declaration

```
public List<string> Sequence { get; set; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Class IdentityChangeRequest

Used to request identity creation or override.

Inheritance

object
IdentityChangeRequest

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Dtos

Assembly: FCli.dll

Syntax

```
public class IdentityChangeRequest
```

Properties

Aliases

Declaration

```
public List<string> Aliases { get; set; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Email

Declaration

```
public string Email { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Name

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Password

Declaration

```
public string Password { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

ToContact()

Declaration

```
public Contact ToContact()
```

Returns

TYPE	DESCRIPTION
Contact	

Class SendEmailRequest

Necessary info to send an email.

Inheritance

object
SendEmailRequest

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Dtos

Assembly: FCli.dll

Syntax

```
public class SendEmailRequest
```

Properties

Body

Declaration

```
public string Body { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

ReceiverEmail

Declaration

```
public string ReceiverEmail { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

ReceiverName

Declaration

```
public string ReceiverName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Subject

Declaration

```
public string Subject { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Namespace FCli.Models.Identity

Classes

Contact

Represents a known user in the identity storage.

IdentityStorage

Object representation of the identity storage.

RootUser

Represents a root user in the identity storage.

Class Contact

Represents a known user in the identity storage.

Inheritance

object
Contact
RootUser

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Identity

Assembly: FCli.dll

Syntax

```
public class Contact
```

Properties

Aliases

Declaration

```
public List<string> Aliases { get; set; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Email

Declaration

```
public string Email { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Name

Declaration

```
public string Name { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

ToChangeRequest()

Declaration

```
public IdentityChangeRequest ToChangeRequest()
```

Returns

TYPE	DESCRIPTION
IdentityChangeRequest	

Class IdentityStorage

Object representation of the identity storage.

Inheritance

object
IdentityStorage

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Identity

Assembly: FCli.dll

Syntax

```
[JsonSerializable(typeof(IdentityStorage))]
public class IdentityStorage
```

Fields

StaticCheckSum

Declaration

```
public static readonly string StaticCheckSum
```

Field Value

TYPE	DESCRIPTION
string	

Properties

CheckSum

Declaration

```
public string CheckSum { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Contacts

Declaration

```
public List<Contact> Contacts { get; set; }
```

Property Value

TYPE	DESCRIPTION
List<Contact>	

PassFileName

Declaration

```
public string PassFileName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

RootUser

Declaration

```
public RootUser RootUser { get; set; }
```

Property Value

TYPE	DESCRIPTION
RootUser	

Class RootUser

Represents a root user in the identity storage.

Inheritance

object
Contact
RootUser

Inherited Members

Contact.Name
Contact.Email
Contact.Aliases
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Models.Identity

Assembly: FCli.dll

Syntax

```
public class RootUser : Contact
```

Remarks

Used as sender for emails.

Constructors

RootUser()

Declaration

```
public RootUser()
```

Properties

Password

Declaration

```
public string Password { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

IsRoot(string)

Declaration

```
public bool IsRoot(string selector)
```

Parameters

TYPE	NAME	DESCRIPTION
string	selector	

Returns

TYPE	DESCRIPTION
bool	

ToChangeRequest()

Declaration

```
public IdentityChangeRequest ToChangeRequest()
```

Returns

TYPE	DESCRIPTION
IdentityChangeRequest	

Namespace FCli.Models.Types

Enums

CommandType

Types that describe command execution.

ShellType

Contains designators for all known shell types.

ToolType

Unique identifiers for tools.

Enum CommandType

Types that describe command execution.

Namespace: FCli.Models.Types

Assembly: FCli.dll

Syntax

```
public enum CommandType
```

Fields

NAME	DESCRIPTION
Directory	
Executable	
Group	
None	
Script	
Website	

Enum ShellType

Contains designators for all known shell types.

Namespace: FCli.Models.Types

Assembly: FCli.dll

Syntax

```
public enum ShellType
```

Fields

NAME	DESCRIPTION
Bash	
Cmd	
Fish	
None	
Powershell	

Enum ToolType

Unique identifiers for tools.

Namespace: FCli.Models.Types

Assembly: FCli.dll

Syntax

```
public enum ToolType
```

Fields

NAME	DESCRIPTION
Add	
Change	
Config	
Group	
Identity	
List	
Mail	
None	
Primes	
Remove	
Run	

Namespace FCli.Services

Classes

ArgsParser

Transforms array of args to `Args` object.

CombinedMailer

Uses both SMTP and IMAP to manage mail.

SystemSpecificFactory

Command factory implementation that recognizes user's OS.

ToolExecutor

Generic implementation of ToolExecutor.

Class ArgsParser

Transforms array of args to `Args` object.

Inheritance

object
Args
ArgsParser

Implements

IArgsParser

Inherited Members

Args.Selector
Args.Arg
Args.Flags
Args.None
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services
Assembly: FCli.dll

Syntax

```
public class ArgsParser : Args, IArgsParser
```

Remarks

Parses command line args differently if they are flat.

Constructors

ArgsParser(ICommandLineFormatter, IResources)

Declaration

```
public ArgsParser(ICommandLineFormatter formatter, IResources resources)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	

Methods

ParseArgs(string[])

Attempts to parse given command line args.

Declaration

```
public Args ParseArgs(string[] args)
```

Parameters

TYPE	NAME	DESCRIPTION
string[]	args	Array of command line arguments.

Returns

TYPE	DESCRIPTION
Args	Parsed `Args` object.

Remarks

Splits args regardless of their organization in the `args` array.

Recognizes strings wrapped in `""` or `''` as one argument.

Each string starting with `--` is treated as a frag key.

One string after a flag key without a starter is treated as a frag argument.

Exceptions

TYPE	CONDITION
ArgumentException	If finds more then one arg and selector.

Implements

IArgsParser

Class CombinedMailer

Uses both SMTP and IMAP to manage mail.

Inheritance

object

CombinedMailer

Implements

IMailer

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services

Assembly: FCli.dll

Syntax

```
public class CombinedMailer : IMailer
```

Constructors

CombinedMailer(ICommandLineFormatter, IResources, IIdentityManager)

Declaration

```
public CombinedMailer(ICommandLineFormatter formatter, IResources resources, IIdentityManager identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IIdentityManager	identity	

Methods

DeleteMessageAsync(int)

Use IMAP client to identify and delete message.

Declaration

```
public Task DeleteMessageAsync(int index)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
int	index	Message identifier.

Returns

TYPE	DESCRIPTION
Task	

ListHeadersAsync(int)

Uses IMAP client to read given amount of emails from the default profile.

Declaration

```
public Task<List<EmailHeaderResponse>> ListHeadersAsync(int amount)
```

Parameters

TYPE	NAME	DESCRIPTION
int	amount	Amount of messages from the top.

Returns

TYPE	DESCRIPTION
Task<List<EmailHeaderResponse>>	List of message headers.

Exceptions

TYPE	CONDITION
IdentityException	If authentication failed.
MailException	If list failed.

ReadMessageAsync(int)

Uses IMAP client to read the contents of the email with given index.

Declaration

```
public Task<EmailMessageResponse> ReadMessageAsync(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	Email identifier.

Returns

TYPE	DESCRIPTION
Task<EmailMessageResponse>	Email response.

Exceptions

TYPE	CONDITION
IdentityException	If authentication failed.
MailException	If read failed.

SendMessageAsync(SendEmailRequest)

Uses SMTP client to send and email to the given address

Declaration

```
public Task SendMessageAsync(SendEmailRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
SendEmailRequest	request	

Returns

TYPE	DESCRIPTION
Task	

Exceptions

TYPE	CONDITION
IdentityException	If authentication failed.
MailException	If send failed.

Implements

IMailer

Class SystemSpecificFactory

Command factory implementation that recognizes user's OS.

Inheritance

object
SystemSpecificFactory

Implements

ICommandFactory

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services

Assembly: FCli.dll

Syntax

```
public class SystemSpecificFactory : ICommandFactory
```

Remarks

Supports Windows and Linux operating systems.

Constructors

SystemSpecificFactory(ICommandLoader, ICommandLineFormatter, IResources)

Declaration

```
public SystemSpecificFactory(ICommandLoader commandLoader, ICommandLineFormatter formatter, IResources
resources)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLoader	commandLoader	
ICommandLineFormatter	formatter	
IResources	resources	

Methods

Construct(CommandAlterRequest)

Should construct new command from the given template.

Declaration

```
public Command Construct(CommandAlterRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
CommandAlterRequest	request	

Returns

TYPE	DESCRIPTION
Command	

Construct(string)

Loads command from storage and reconstructs it using OS specific templates.

Declaration

```
public Command Construct(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	Command selector.

Returns

TYPE	DESCRIPTION
Command	Command constructed form the storage.

Exceptions

TYPE	CONDITION
InvalidOperationException	If given name is unknown.

ConstructGroup(GroupAlterRequest)

Generates a group of sequentially executed commands.

Declaration

```
public Group ConstructGroup(GroupAlterRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
GroupAlterRequest	request	Request model for the desired group.

Returns

TYPE	DESCRIPTION
Group	Constructed group object.

Implements

ICommandFactory

Class ToolExecutor

Generic implementation of ToolExecutor.

Inheritance

object
ToolExecutor

Implements

IToolExecutor

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services

Assembly: FCli.dll

Syntax

```
public class ToolExecutor : IToolExecutor
```

Constructors

ToolExecutor(ILogger<ToolExecutor>, IEnumerable<ITool>)

Declaration

```
public ToolExecutor(ILogger<ToolExecutor> logger, IEnumerable<ITool> tools)
```

Parameters

TYPE	NAME	DESCRIPTION
ILogger<ToolExecutor>	logger	
IEnumerable<ITool>	tools	

Methods

Execute(Args, ToolType)

Execute tool from given type and arg.

Declaration

```
public void Execute(Args args, ToolType type)
```

Parameters

TYPE	NAME	DESCRIPTION
Args	args	Tool argument.

TYPE	NAME	DESCRIPTION
ToolType	type	Tool type to execute.

Exceptions

TYPE	CONDITION
CriticalException	If tool selection fails.

ParseType(Args)

Parses given args and returns tool type if recognized.

Declaration

```
public ToolType ParseType(Args args)
```

Parameters

TYPE	NAME	DESCRIPTION
Args	args	Args to analyze.

Returns

TYPE	DESCRIPTION
ToolType	Tool type or None.

Implements

IToolExecutor

Namespace FCli.Services.Abstractions

Classes

IConfig.CommandDescriptor

IConfig.FormatterDescriptor

IConfig.ShellDescriptor

Interfaces

IArgsParser

Tries to parse array of command line args in the most appropriate way.

ICommandFactory

Describes generic factory that constructs commands from templates.

ICommandLineFormatter

Describes a service that writes formatted text into the console.

ICommandLoader

Describes generic class that performs CRUD operations on `Command`.

IConfig

Abstraction for fcli configuration. Includes both static and dynamic configs.

IEncryptor

Uses certain encryption algorithm to translate sensitive data.

IIdentityManager

Manages identity storage. Used by mail subsystem.

IMailer

Represents mailing subsystem. Allows to send, read and delete mail.

IResources

Wrapper for all interactions with app resources.

ITool

Represents a command line tool.

IToolDescriptor

Represents main tool info.

IToolExecutor

Describes class that parses command args and executes tools.

Interface IArgsParser

Tries to parse array of command line args in the most appropriate way.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IArgsParser
```

Methods

ParseArgs(string[])

Should correctly parse raw command line args into fcli specific args abstraction object.

Declaration

```
Args ParseArgs(string[] args)
```

Parameters

TYPE	NAME	DESCRIPTION
string[]	args	Command line args.

Returns

TYPE	DESCRIPTION
Args	Parsed Args object.

Interface ICommandFactory

Describes generic factory that constructs commands from templates.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface ICommandFactory
```

Methods

Construct(CommandAlterRequest)

Should construct new command from the given template.

Declaration

```
Command Construct(CommandAlterRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
CommandAlterRequest	request	

Returns

TYPE	DESCRIPTION
Command	

Construct(string)

Should load a command template from the storage and construct it.

Declaration

```
Command Construct(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	

Returns

TYPE	DESCRIPTION
Command	

ConstructGroup(GroupAlterRequest)

Should generate a group of commands.

Declaration

```
Group ConstructGroup(GroupAlterRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
GroupAlterRequest	request	

Returns

TYPE	DESCRIPTION
Group	

Interface ICommandLineFormatter

Describes a service that writes formatted text into the console.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface ICommandLineFormatter
```

Remarks

Has some default implementations.

Methods

DisplayError(string?, string?)

Errors the given message to console.

Declaration

```
void DisplayError(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DisplayInfo(string?, string?)

Formats string as an Info message and displays it to the console.

Declaration

```
void DisplayInfo(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DisplayMessage(string?)

Simply prints out the message into the console.

Declaration

```
void DisplayMessage(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	String to echo out.

DisplayProgressMessage(string?)

Displays messages while progress is running.

Declaration

```
void DisplayProgressMessage(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	To display.

DisplayWarning(string?, string?)

Displays the string in the console as a Warning.

Declaration

```
void DisplayWarning(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DrawProgressAsync(CancellationToken)

Draws progress graphic to console.

Declaration

```
Task DrawProgressAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	Used to stop progress drawing.

Returns

TYPE	DESCRIPTION
Task	

EchoGreeting()

Should echo logo and basic helpful info about fallen-cli.

Declaration

```
void EchoGreeting()
```

EchoHelp()

Should display even more helpful info then the basic greeting.

Declaration

```
void EchoHelp()
```

EchoLogo()

Everyone needs one.

Declaration

```
void EchoLogo()
```

EchoNameAndVersion()

Writes to the console assembly name and version.

Declaration

```
void EchoNameAndVersion()
```

ReadUserInput(string?, bool)

Prints out a formatted preface and then reads user's input.

Declaration

```
string? ReadUserInput(string? preface, bool hideInput = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	preface	The string that is put before user input.
bool	hideInput	

Returns

TYPE	DESCRIPTION
string	User input.

Interface ICommandLoader

Describes generic class that performs CRUD operations on `Command`.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface ICommandLoader
```

Methods

CommandExists(string)

Should check if command is present in the storage and return `false` if not.

Declaration

```
bool CommandExists(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	

Returns

TYPE	DESCRIPTION
bool	

DeleteCommand(string)

Should attempt to delete a command from the given name and throw if that command doesn't exist.

Declaration

```
void DeleteCommand(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	

LoadCommand(string)

Should attempt loading a command from the given name and return `null` if fails.

Declaration

```
Command? LoadCommand(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	

Returns

TYPE	DESCRIPTION
Command	

LoadCommands()

Should attempt loading the whole command storage and return `null` if the storage is empty or doesn't exist yet.

Declaration

```
List<Command>? LoadCommands()
```

Returns

TYPE	DESCRIPTION
List<Command>	

SaveCommand(Command)

Should save the given command to storage.

Declaration

```
void SaveCommand(Command command)
```

Parameters

TYPE	NAME	DESCRIPTION
Command	command	

Interface IConfig

Abstraction for fcli configuration. Includes both static and dynamic configs.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IConfig
```

Properties

AppFolderName

Name for the root app folder.

Declaration

```
string AppFolderName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

AppFolderPath

Root app path.

Declaration

```
string AppFolderPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

ConfigFileName

Name of the file that stores this object.

Declaration

```
string ConfigFileName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

ConfigFilePath

Path to the configuration file.

Declaration

```
string ConfigFilePath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Formatter

Returns current selected command line formatter.

Declaration

```
IConfig.FormatterDescriptor Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
IConfig.FormatterDescriptor	

IdentityFileName

Name of the file that contains all users.

Declaration

```
string IdentityFileName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

IdentityFilePath

Path to the file with user data.

Declaration

```
string IdentityFilePath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

KnownCommands

List of all known flags that describe command flavors.

Declaration

```
List<IConfig.CommandDescriptor> KnownCommands { get; }
```

Property Value

TYPE	DESCRIPTION
List<IConfig.CommandDescriptor>	

Remarks

Value consists of Command type and a flag that is true if this command executed in the shell.

KnownFormatters

Return pairs of formatter-selector and formatter-type.

Declaration

```
List<IConfig.FormatterDescriptor> KnownFormatters { get; }
```

Property Value

TYPE	DESCRIPTION
List<IConfig.FormatterDescriptor>	

KnownLocales

List of all known locales.

Declaration

```
List<string> KnownLocales { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

KnownShells

List of all known shells designators with respective types.

Declaration

```
List<IConfig.ShellDescriptor> KnownShells { get; }
```

Property Value

TYPE	DESCRIPTION
List<IConfig.ShellDescriptor>	

Remarks

Value consists of Shell type and a specific shell file extension.

KnownTools

List all known fcli tools.

Declaration

```
List<IToolDescriptor> KnownTools { get; }
```

Property Value

TYPE	DESCRIPTION
List<IToolDescriptor>	

Locale

Returns current culture decided by the user.

Declaration

```
string Locale { get; }
```

Property Value

TYPE	DESCRIPTION
string	

LogsFileTemplate

Template for log file names.

Declaration

```
string LogsFileTemplate { get; }
```

Property Value

TYPE	DESCRIPTION
string	

LogsFolderName

Name for the folder that contains logs.

Declaration

```
string LogsFolderName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

LogsPath

Path to the logs template.

Declaration

```
string LogsPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

PassphraseFile

The path to the file that temporarily stores the passphrase.

Declaration

```
string PassphraseFile { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Remarks

May be bad path.

Salt

Used to offset encryption. Generated automatically.

Declaration

```
byte[] Salt { get; }
```

Property Value

TYPE	DESCRIPTION
byte[]	

StorageFileName

Name for the command storage file.

Declaration

```
string StorageFileName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

StorageFilePath

Path to the command storage file.

Declaration

```
string StorageFilePath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

StringsResourceLocation

Path to the Strings resource file.

Declaration

```
string StringsResourceLocation { get; }
```

Property Value

TYPE	DESCRIPTION
string	

UseEncryption

Specifies if fcli should encrypt user data.

Declaration

```
bool UseEncryption { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

ChangeAppFolder(DirectoryInfo?)

Should change default files' location.

Declaration

```
void ChangeAppFolder(DirectoryInfo? directory)
```

Parameters

TYPE	NAME	DESCRIPTION
DirectoryInfo	directory	

ChangeEncryption(bool)

Should sets new value for the UseEncryption flag.

Declaration

```
void ChangeEncryption(bool encrypt)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	encrypt	

ChangeFormatter(FormatterDescriptor)

Should change the default console formatter.

Declaration

```
void ChangeFormatter(IConfig.FormatterDescriptor formatter)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig.FormatterDescriptor	formatter	New formatter.

ChangeLocale(string)

Should change the locale in the config.

Declaration

```
void ChangeLocale(string locale)
```

Parameters

TYPE	NAME	DESCRIPTION
string	locale	New locale.

ChangePassphraseFile(string)

Should change last file name that stored the passphrase.

Declaration

```
void ChangePassphraseFile(string filename)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filename	New file name.

ChangeSalt()

Should regenerate encryption salt.

Declaration

```
void ChangeSalt()
```

LoadConfig()

Should save this config as is to storage.

Declaration

```
void LoadConfig()
```

PurgeConfig()

Should delete config file.

Declaration

```
void PurgeConfig()
```

SaveConfig()

Saves current config to storage.

Declaration

```
void SaveConfig()
```

Class IConfig.CommandDescriptor

Inheritance

object
IConfig.CommandDescriptor

Implements

IEquatable<IConfig.CommandDescriptor>

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public record IConfig.CommandDescriptor : IEquatable<IConfig.CommandDescriptor>
```

Constructors

CommandDescriptor(string, CommandType, bool, string?)

Declaration

```
public CommandDescriptor(string Selector, CommandType Type, bool IsShell, string? FileExtension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	Selector	
CommandType	Type	
bool	IsShell	
string	FileExtension	

Properties

FileExtension

Declaration

```
public string? FileExtension { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

IsShell

Declaration

```
public bool IsShell { get; init; }
```

Property Value

TYPE	DESCRIPTION
bool	

Selector

Declaration

```
public string Selector { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Type

Declaration

```
public CommandType Type { get; init; }
```

Property Value

TYPE	DESCRIPTION
CommandType	

Implements

IEquatable<T>

Class IConfig.FormatterDescriptor

Inheritance

object
IConfig.FormatterDescriptor

Implements

IEquatable<IConfig.FormatterDescriptor>

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public record IConfig.FormatterDescriptor : IEquatable<IConfig.FormatterDescriptor>
```

Constructors

FormatterDescriptor(string, Type)

Declaration

```
public FormatterDescriptor(string Selector, Type Type)
```

Parameters

TYPE	NAME	DESCRIPTION
string	Selector	
Type	Type	

Properties

Selector

Declaration

```
public string Selector { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Type

Declaration

```
public Type Type { get; init; }
```

Property Value

TYPE	DESCRIPTION
Type	

Implements

IEquatable<T>

Class IConfig.ShellDescriptor

Inheritance

object
IConfig.ShellDescriptor

Implements

IEquatable<IConfig.ShellDescriptor>

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public record IConfig.ShellDescriptor : IEquatable<IConfig.ShellDescriptor>
```

Constructors

ShellDescriptor(string, ShellType, string)

Declaration

```
public ShellDescriptor(string Selector, ShellType Type, string FileExtension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	Selector	
ShellType	Type	
string	FileExtension	

Properties

FileExtension

Declaration

```
public string FileExtension { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Selector

Declaration

```
public string Selector { get; init; }
```

Property Value

TYPE	DESCRIPTION
string	

Type

Declaration

```
public ShellType Type { get; init; }
```

Property Value

TYPE	DESCRIPTION
ShellType	

Implements

IEquatable<T>

Interface IEncryptor

Uses certain encryption algorithm to translate sensitive data.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IEncryptor
```

Methods

Decrypt(string, string)

Decrypts given string.

Declaration

```
string Decrypt(string encrypted64, string passphrase)
```

Parameters

TYPE	NAME	DESCRIPTION
string	encrypted64	
string	passphrase	Passphrase used for decryption.

Returns

TYPE	DESCRIPTION
string	Decrypted string.

Encrypt(string, string)

Encrypts given string.

Declaration

```
string Encrypt(string unencrypted, string passphrase)
```

Parameters

TYPE	NAME	DESCRIPTION
string	unencrypted	String to encrypt.
string	passphrase	Passphrase used for encryption.

Returns

TYPE	DESCRIPTION
string	Encrypted string.

Interface IIdentityManager

Manages identity storage. Used by mail subsystem.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IIdentityManager
```

Methods

ContactExists(string)

Checks if storage contains given name or alias.

Declaration

```
bool ContactExists(string nameOrAlias)
```

Parameters

TYPE	NAME	DESCRIPTION
string	nameOrAlias	Selector to check.

Returns

TYPE	DESCRIPTION
bool	True if found.

DeleteContact(string)

Removes given user from storage.

Declaration

```
void DeleteContact(string userName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	userName	Selector to delete.

GetRootUser()

Loads last root user profile.

Declaration

```
RootUser GetRootUser()
```

Returns

TYPE	DESCRIPTION
RootUser	Current root profile.

LoadContact(string)

Retrieves a contact with given selector.

Declaration

```
Contact? LoadContact(string nameOrAlias)
```

Parameters

TYPE	NAME	DESCRIPTION
string	nameOrAlias	Selector to search.

Returns

TYPE	DESCRIPTION
Contact	Loaded contact or null if not found.

LoadContacts()

Loads known contacts form the storage.

Declaration

```
List<Contact>? LoadContacts()
```

Returns

TYPE	DESCRIPTION
List<Contact>	List of contacts or null if nothing is stored.

StoreContact(Contact)

Persists new contact.

Declaration

```
void StoreContact(Contact user)
```

Parameters

TYPE	NAME	DESCRIPTION
Contact	user	Contact profile.

UpdateRootUser(IdentityChangeRequest)

Changes root user profile.

Declaration

```
void UpdateRootUser(IdentityChangeRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
IdentityChangeRequest	request	

Interface IMailer

Represents mailing subsystem. Allows to send, read and delete mail.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IMailer
```

Methods

DeleteMessageAsync(int)

Deletes email with given index from the provider.

Declaration

```
Task DeleteMessageAsync(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	Deleting email identifier.

Returns

TYPE	DESCRIPTION
Task	

ListHeadersAsync(int)

List given amount of email headers from the end.

Declaration

```
Task<List<EmailHeaderResponse>> ListHeadersAsync(int amount)
```

Parameters

TYPE	NAME	DESCRIPTION
int	amount	Amount of mail to load.

Returns

TYPE	DESCRIPTION
Task<List<EmailHeaderResponse>>	Loaded headers.

ReadMessageAsync(int)

Loads full message with the given index.

Declaration

```
Task<EmailMessageResponse> ReadMessageAsync(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	Email identifier.

Returns

TYPE	DESCRIPTION
Task<EmailMessageResponse>	Loaded email.

SendMessageAsync(SendEmailRequest)

Performs send email task.

Declaration

```
Task SendMessageAsync(SendEmailRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
SendEmailRequest	request	Necessary email information.

Returns

TYPE	DESCRIPTION
Task	

Interface IResources

Wrapper for all interactions with app resources.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IResources
```

Methods

GetLocalizedString(string)

Should load the string from the resource files according to the user's local culture.

Declaration

```
string GetLocalizedString(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	Name of the resource string .

Returns

TYPE	DESCRIPTION
string	Loaded string.

Interface ITool

Represents a command line tool.

Inherited Members

IToolDescriptor.Name
IToolDescriptor.Description
IToolDescriptor.Selectors
IToolDescriptor.Type

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface ITool : IToolDescriptor
```

Methods

Execute(string, IEnumerable<Flag>)

Performs main tool logic.

Declaration

```
void Execute(string arg, IEnumerable<Flag> flags)
```

Parameters

TYPE	NAME	DESCRIPTION
string	arg	Tool's arg.
IEnumerable<Flag>	flags	Tool's flags

Interface IToolDescriptor

Represents main tool info.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IToolDescriptor
```

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Name

Toll's command line selector.

Declaration

```
string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Selectors

Known aliases for the selector of the tool.

Declaration

```
List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Type

Unique descriptor for the tool.

Declaration

```
ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Interface IToolExecutor

Describes class that parses command args and executes tools.

Namespace: FCli.Services.Abstractions

Assembly: FCli.dll

Syntax

```
public interface IToolExecutor
```

Methods

Execute(Args, ToolType)

Should execute tool of given type with given args.

Declaration

```
void Execute(Args args, ToolType type)
```

Parameters

TYPE	NAME	DESCRIPTION
Args	args	
ToolType	type	

ParseType(Args)

Should determine tool type form args or return None.

Declaration

```
ToolType ParseType(Args args)
```

Parameters

TYPE	NAME	DESCRIPTION
Args	args	

Returns

TYPE	DESCRIPTION
ToolType	

Namespace FCli.Services.Config

Classes

DynamicConfig

Encapsulates the part of the config that can be changed by the user.

StaticConfig

Encapsulates the static layer of the config.

Class DynamicConfig

Encapsulates the part of the config that can be changed by the user.

Inheritance

object
StaticConfig
DynamicConfig

Implements

IConfig

Inherited Members

StaticConfig.StorageFileName
StaticConfig.StorageFilePath
StaticConfig.ConfigFileName
StaticConfig.ConfigFilePath
StaticConfig.IdentityFileName
StaticConfig.IdentityFilePath
StaticConfig.AppFolderName
StaticConfig.AppFolderPath
StaticConfig.LogsFileTemplate
StaticConfig.LogsFolderName
StaticConfig.LogsPath
StaticConfig.PassphraseFile
StaticConfig.Locale
StaticConfig.UseEncryption
StaticConfig.Salt
StaticConfig.Formatter
StaticConfig.KnownLocales
StaticConfig.KnownFormatters
StaticConfig.KnownTools
StaticConfig.KnownCommands
StaticConfig.KnownShells
StaticConfig.StringsResourceLocation
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Config

Assembly: FCli.dll

Syntax

```
public class DynamicConfig : StaticConfig, IConfig
```

Constructors

DynamicConfig()

Declaration

```
public DynamicConfig()
```

Methods

ChangeAppFolder(DirectoryInfo?)

Changes app default files location.

Declaration

```
public override void ChangeAppFolder(DirectoryInfo? directory)
```

Parameters

TYPE	NAME	DESCRIPTION
DirectoryInfo	directory	New app path.

Overrides

StaticConfig.ChangeAppFolder(DirectoryInfo?)

ChangeEncryption(bool)

Changes encryption.

Declaration

```
public override void ChangeEncryption(bool encrypt)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	encrypt	True if encrypt.

Overrides

StaticConfig.ChangeEncryption(bool)

ChangeFormatter(FormatterDescriptor)

Changes formatter without validation.

Declaration

```
public override void ChangeFormatter(IConfig.FormatterDescriptor formatter)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig.FormatterDescriptor	formatter	New formatter.

Overrides

StaticConfig.ChangeFormatter(IConfig.FormatterDescriptor)

ChangeLocale(string)

Changes locale without validation.

Declaration

```
public override void ChangeLocale(string locale)
```

Parameters

TYPE	NAME	DESCRIPTION
string	locale	New locale.

Overrides

StaticConfig.ChangeLocale(string)

ChangePassphraseFile(string)

Changes last passphrase file.

Declaration

```
public override void ChangePassphraseFile(string filename)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filename	

Overrides

StaticConfig.ChangePassphraseFile(string)

ChangeSalt()

Regenerates salt.

Declaration

```
public override void ChangeSalt()
```

Overrides

StaticConfig.ChangeSalt()

LoadConfig()

Loads user config from storage and deserializes it.

Declaration

```
public override void LoadConfig()
```

Overrides

StaticConfig.LoadConfig()

PurgeConfig()

Deletes identity and config files.

Declaration

```
public override void PurgeConfig()
```

Overrides

StaticConfig.PurgeConfig()

SaveConfig()

Serializes this object to json.

Declaration

```
public override void SaveConfig()
```

Overrides

StaticConfig.SaveConfig()

Implements

IConfig

Class StaticConfig

Encapsulates the static layer of the config.

Inheritance

object
StaticConfig
DynamicConfig

Implements

IConfig

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Config

Assembly: FCli.dll

Syntax

```
public abstract class StaticConfig : IConfig
```

Properties

AppFolderName

Name for the root app folder.

Declaration

```
public string AppFolderName { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

AppFolderPath

Root app path.

Declaration

```
public string AppFolderPath { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

ConfigFileName

Name of the file that stores this object.

Declaration

```
public string ConfigFileName { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

ConfigFilePath

Path to the configuration file.

Declaration

```
public string ConfigFilePath { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

Formatter

Returns current selected command line formatter.

Declaration

```
public IConfig.FormatterDescriptor Formatter { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
IConfig.FormatterDescriptor	

IdentityFileName

Name of the file that contains all users.

Declaration

```
public string IdentityFileName { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

IdentityFilePath

Path to the file with user data.

Declaration

```
public string IdentityFilePath { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

KnownCommands

List of all known flags that describe command flavors.

Declaration

```
public List<IConfig.CommandDescriptor> KnownCommands { get; }
```

Property Value

TYPE	DESCRIPTION
List<IConfig.CommandDescriptor>	

Remarks

Value consists of Command type and a flag that is true if this command executed in the shell.

KnownFormatters

Return pairs of formatter-selector and formatter-type.

Declaration

```
public List<IConfig.FormatterDescriptor> KnownFormatters { get; }
```

Property Value

TYPE	DESCRIPTION
List<IConfig.FormatterDescriptor>	

KnownLocales

List of all known locales.

Declaration

```
public List<string> KnownLocales { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

KnownShells

List of all known shells designators with respective types.

Declaration

```
public List<IConfig.ShellDescriptor> KnownShells { get; }
```

Property Value

TYPE	DESCRIPTION
List<IConfig.ShellDescriptor>	

Remarks

Value consists of Shell type and a specific shell file extension.

KnownTools

List all known fcli tools.

Declaration

```
public List<IToolDescriptor> KnownTools { get; }
```

Property Value

TYPE	DESCRIPTION
List<IToolDescriptor>	

Locale

Returns current culture decided by the user.

Declaration

```
public string Locale { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

LogsFileTemplate

Template for log file names.

Declaration

```
public string LogsFileTemplate { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

LogsFolderName

Name for the folder that contains logs.

Declaration

```
public string LogsFolderName { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

LogsPath

Path to the logs template.

Declaration

```
public string LogsPath { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

PassphraseFile

The path to the file that temporarily stores the passphrase.

Declaration

```
public string PassphraseFile { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

Remarks

May be bad path.

Salt

Used to offset encryption. Generated automatically.

Declaration

```
public byte[] Salt { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
byte[]	

StorageFileName

Name for the command storage file.

Declaration

```
public string StorageFileName { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

StorageFilePath

Path to the command storage file.

Declaration

```
public string StorageFilePath { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

StringsResourceLocation

Path to the Strings resource file.

Declaration

```
public string StringsResourceLocation { get; }
```

Property Value

TYPE	DESCRIPTION
string	

UseEncryption

Specifies if fcli should encrypt user data.

Declaration

```
public bool UseEncryption { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

ChangeAppFolder(DirectoryInfo?)

Should change default files' location.

Declaration

```
public abstract void ChangeAppFolder(DirectoryInfo? directory)
```

Parameters

TYPE	NAME	DESCRIPTION
DirectoryInfo	directory	

ChangeEncryption(bool)

Should sets new value for the UseEncryption flag.

Declaration

```
public abstract void ChangeEncryption(bool encrypt)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	encrypt	

ChangeFormatter(FormatterDescriptor)

Should change the default console formatter.

Declaration

```
public abstract void ChangeFormatter(IConfig.FormatterDescriptor formatter)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig.FormatterDescriptor	formatter	New formatter.

ChangeLocale(string)

Should change the locale in the config.

Declaration

```
public abstract void ChangeLocale(string locale)
```

Parameters

TYPE	NAME	DESCRIPTION
string	locale	New locale.

ChangePassphraseFile(string)

Should change last file name that stored the passphrase.

Declaration

```
public abstract void ChangePassphraseFile(string filename)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filename	New file name.

ChangeSalt()

Should regenerate encryption salt.

Declaration

```
public abstract void ChangeSalt()
```

LoadConfig()

Should save this config as is to storage.

Declaration

```
public abstract void LoadConfig()
```

PurgeConfig()

Should delete config file.

Declaration

```
public abstract void PurgeConfig()
```

SaveConfig()

Saves current config to storage.

Declaration

```
public abstract void SaveConfig()
```

Implements

IConfig

Namespace FCli.Services.Data

Classes

JsonLoader

Command loader that uses json format to store and read commands.

StringResources

Uses ResourceManager class to load strings from app resources.

Class JsonLoader

Command loader that uses json format to store and read commands.

Inheritance

object
JsonLoader

Implements

ICommandLoader

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Data

Assembly: FCli.dll

Syntax

```
public class JsonLoader : ICommandLoader
```

Remarks

Storage file location is specified through the `DynamicConfig` properties.

Constructors

JsonLoader(IConfig)

Declaration

```
public JsonLoader(IConfig config)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig	config	

Methods

CommandExists(string)

Checks if command exists.

Declaration

```
public bool CommandExists(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the command.

Returns

TYPE	DESCRIPTION
bool	`true` if command exists, `false` if not.

Remarks

Uses buffer and attempts to `LoadCommands` if it is empty.

DeleteCommand(string)

Attempts to delete given command.

Declaration

```
public void DeleteCommand(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the command to delete.

Remarks

Refreshes the whole storage file if deletion is successful.

Exceptions

TYPE	CONDITION
ArgumentException	If command doesn't exist.

LoadCommand(string)

Attempts to load a command.

Declaration

```
public Command? LoadCommand(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the command.

Returns

TYPE	DESCRIPTION
Command	Loaded command, or `null` if it wasn't found.

Remarks

Checks buffer and attempts to load commands otherwise.

LoadCommands()

Loads all commands from storage file specified in `DynamicConfig`.

Declaration

```
public List<Command>? LoadCommands()
```

Returns

TYPE	DESCRIPTION
List<Command>	Loaded command buffer, or `null` if load fails.

Remarks

Populates commands buffer if those commands were successfully loaded and parsed.

Uses System.Text.Json serializer to work with json string.

Exceptions

TYPE	CONDITION
CriticalException	If command deserialization fails.

SaveCommand(Command)

Saves given command to storage.

Declaration

```
public void SaveCommand(Command command)
```

Parameters

TYPE	NAME	DESCRIPTION
Command	command	Command object to save.

Remarks

Refreshes the whole storage file upon saving the command.

Implements

ICommandLoader

Class StringResources

Uses ResourceManager class to load strings from app resources.

Inheritance

object

StringResources

Implements

IResources

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Data

Assembly: FCli.dll

Syntax

```
public class StringResources : IResources
```

Constructors

StringResources(IConfig)

Declaration

```
public StringResources(IConfig config)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig	config	

Methods

GetLocalizedString(string)

Uses resource manager to extract string according to user's locale.

Declaration

```
public string GetLocalizedString(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	String name in the Strings resource file.

Returns

TYPE	DESCRIPTION
string	Loaded string.

Implements

IResources

Namespace FCli.Services.Data.Identity

Classes

EncryptedIdentityManager

Extension of PlainManager that uses encryption provider to safely store identities.

PlainIdentityManager

Uses json to store identities. Cashes values.

Class EncryptedIdentityManager

Extension of PlainManager that uses encryption provider to safely store identities.

Inheritance

object
PlainIdentityManager
EncryptedIdentityManager

Implements

IIdentityManager

Inherited Members

PlainIdentityManager.Config
PlainIdentityManager.IdentityCashe
PlainIdentityManager.LoadContacts()
PlainIdentityManager.LoadContact(string)
PlainIdentityManager.ContactExists(string)
PlainIdentityManager.GetRootUser()
PlainIdentityManager.UpdateRootUser(IdentityChangeRequest)
PlainIdentityManager.StoreContact(Contact)
PlainIdentityManager.DeleteContact(string)
PlainIdentityManager.GetUserFromAlias(IdentityStorage, string)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Data.Identity

Assembly: FCli.dll

Syntax

```
public class EncryptedIdentityManager : PlainIdentityManager, IIdentityManager
```

Constructors

EncryptedIdentityManager(ICommandLineFormatter, IResources, IConfig, IEncryptor)

Declaration

```
public EncryptedIdentityManager(ICommandLineFormatter formatter, IResources resources, IConfig config,
IEncryptor encrypter)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
IEncryptor	encrypter	

Methods

DecryptStorage()

Assumes that storage is encrypted and transforms it to plain.

Declaration

```
public void DecryptStorage()
```

EncryptStorage()

Assumes that storage is plain and encrypts it.

Declaration

```
public void EncryptStorage()
```

FlushStorage(IdentityStorage?)

Encrypted version of storage update.

Declaration

```
protected override void FlushStorage(IdentityStorage? storage)
```

Parameters

TYPE	NAME	DESCRIPTION
IdentityStorage	storage	Identities to store.

Overrides

PlainIdentityManager.FlushStorage(IdentityStorage?)

TryLoadStorage()

Tries to load and decrypt storage.

Declaration

```
protected override IdentityStorage? TryLoadStorage()
```

Returns

TYPE	DESCRIPTION
IdentityStorage	Loaded storage if exists, null otherwise.

Overrides

PlainIdentityManager.TryLoadStorage()

Exceptions

TYPE	CONDITION
IdentityException	If decryption fails.

TYPE	CONDITION
CriticalException	If storage corrupted.

Implements

IIdentityManager

Class PlainIdentityManager

Uses json to store identities. Cashes values.

Inheritance

object
PlainIdentityManager
EncryptedIdentityManager

Implements

IIdentityManager

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Data.Identity

Assembly: FCli.dll

Syntax

```
public class PlainIdentityManager : IIdentityManager
```

Constructors

PlainIdentityManager(IConfig)

Declaration

```
public PlainIdentityManager(IConfig config)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig	config	

Properties

Config

Declaration

```
protected IConfig Config { get; }
```

Property Value

TYPE	DESCRIPTION
IConfig	

IdentityCashe

Declaration

```
protected IdentityStorage? IdentityCashe { get; set; }
```

Property Value

TYPE	DESCRIPTION
IdentityStorage	

Methods

ContactExists(string)

Checks if storage contains given name or alias.

Declaration

```
public bool ContactExists(string nameOrAlias)
```

Parameters

TYPE	NAME	DESCRIPTION
string	nameOrAlias	Selector to check.

Returns

TYPE	DESCRIPTION
bool	True if found.

DeleteContact(string)

Removes given user from storage.

Declaration

```
public void DeleteContact(string userName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	userName	Selector to delete.

FlushStorage(IdentityStorage?)

Refreshes the identity storage.

Declaration

```
protected virtual void FlushStorage(IdentityStorage? storage)
```

Parameters

TYPE	NAME	DESCRIPTION
IdentityStorage	storage	New storage.

GetRootUser()

Loads last root user profile.

Declaration

```
public RootUser GetRootUser()
```

Returns

TYPE	DESCRIPTION
RootUser	Current root profile.

GetUserFromAlias(IdentityStorage?, string)

Checks if user with given selector is present in the storage.

Declaration

```
protected static Contact? GetUserFromAlias(IdentityStorage? storage, string nameOrAlias)
```

Parameters

TYPE	NAME	DESCRIPTION
IdentityStorage	storage	Storage where to look.
string	nameOrAlias	Selector to search.

Returns

TYPE	DESCRIPTION
Contact	Contact if found, null if not.

LoadContact(string)

Retrieves a contact with given selector.

Declaration

```
public Contact? LoadContact(string nameOrAlias)
```

Parameters

TYPE	NAME	DESCRIPTION
string	nameOrAlias	Selector to search.

Returns

TYPE	DESCRIPTION
Contact	Loaded contact or null if not found.

LoadContacts()

Loads known contacts form the storage.

Declaration

```
public List<Contact>? LoadContacts()
```

Returns

TYPE	DESCRIPTION
List<Contact>	List of contacts or null if nothing is stored.

StoreContact(Contact)

Persists new contact.

Declaration

```
public void StoreContact(Contact user)
```

Parameters

TYPE	NAME	DESCRIPTION
Contact	user	Contact profile.

TryLoadStorage()

Tries to load storage if it exists.

Declaration

```
protected virtual IdentityStorage? TryLoadStorage()
```

Returns

TYPE	DESCRIPTION
IdentityStorage	Identity storage, or null if it is not present.

Exceptions

TYPE	CONDITION
CriticalException	If storage is corrupted.

UpdateRootUser(IdentityChangeRequest)

Changes root user profile.

Declaration

```
public void UpdateRootUser(IdentityChangeRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
IdentityChangeRequest	request	

Implements

IIdentityManager

Namespace FCli.Services.Encryption

Classes

AesEncryptor

This class uses AES algorithm to encrypt and decrypt strings.

Class AesEncryptor

This class uses AES algorithm to encrypt and decrypt strings.

Inheritance

object
AesEncryptor

Implements

IEncryptor

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Encryption

Assembly: FCli.dll

Syntax

```
public class AesEncryptor : IEncryptor
```

Constructors

AesEncryptor(IConfig)

Declaration

```
public AesEncryptor(IConfig config)
```

Parameters

TYPE	NAME	DESCRIPTION
IConfig	config	

Methods

Decrypt(string, string)

Converts given encrypted base64 string and decrypts it.

Declaration

```
public string Decrypt(string encrypted64, string passphrase)
```

Parameters

TYPE	NAME	DESCRIPTION
string	encrypted64	Original data.
string	passphrase	Passphrase to decrypt.

Returns

TYPE	DESCRIPTION
string	Decrypted data.

Encrypt(string, string)

Transforms given plain text into encrypted base64 string.

Declaration

```
public string Encrypt(string unencrypted, string passphrase)
```

Parameters

TYPE	NAME	DESCRIPTION
string	unencrypted	Source data.
string	passphrase	Passphrase to encrypt.

Returns

TYPE	DESCRIPTION
string	Encrypted base64 string.

Implements

IEncryptor

Namespace FCli.Services.Format

Classes

InlineFormatter

Command line formatter that uses multiline messages and colors.

PrettyFormatter

Command line formatter that uses multiline messages and colors.

Class InlineFormatter

Command line formatter that uses multiline messages and colors.

Inheritance

object

InlineFormatter

Implements

ICommandLineFormatter

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: FCli.Services.Format

Assembly: FCli.dll

Syntax

```
public class InlineFormatter : ICommandLineFormatter
```

Constructors

InlineFormatter(IResources)

Declaration

```
public InlineFormatter(IResources strings)
```

Parameters

TYPE	NAME	DESCRIPTION
IResources	strings	

Methods

DisplayError(string?, string?)

Formats Error as single line with red caller name and normal message.

Declaration

```
public void DisplayError(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DisplayInfo(string?, string?)

Formats Info as a line starting with green caller name and normal inline message.

Declaration

```
public void DisplayInfo(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DisplayMessage(string?)

Writes the message to the console.

Declaration

```
public void DisplayMessage(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	String to be printed to console.

DisplayProgressMessage(string?)

Writes message cleanly if progress is running.

Declaration

```
public void DisplayProgressMessage(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	To display.

DisplayWarning(string?, string?)

Formats Waring in one line starting with yellow caller name and ending with normal inline message.

Declaration

```
public void DisplayWarning(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DrawProgressAsync(CancellationToken)

Draws progress as simple loading animated message.

Declaration

```
public Task DrawProgressAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	Used to stop the progress.

Returns

TYPE	DESCRIPTION
Task	

EchoGreeting()

Loads basic info from the resources.

Declaration

```
public void EchoGreeting()
```

EchoHelp()

Loads full help page for the entire fallen-cli from the resources.

Declaration

```
public void EchoHelp()
```

ReadUserInput(string?, bool)

Formats input line as a plain single liner.

Declaration

```
public string? ReadUserInput(string? preface, bool hideInput = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	preface	String that is written before input.

TYPE	NAME	DESCRIPTION
bool	hideInput	If true hides user input.

Returns

TYPE	DESCRIPTION
string	User input.

Implements

ICommandLineFormatter

Class PrettyFormatter

Command line formatter that uses multiline messages and colors.

Inheritance

object
PrettyFormatter

Implements

ICommandLineFormatter

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Format

Assembly: FCli.dll

Syntax

```
public class PrettyFormatter : ICommandLineFormatter
```

Constructors

PrettyFormatter(IResources)

Declaration

```
public PrettyFormatter(IResources resources)
```

Parameters

TYPE	NAME	DESCRIPTION
IResources	resources	

Methods

DisplayError(string?, string?)

Formats Error as two lines from red caller name and indented yellow message.

Declaration

```
public void DisplayError(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DisplayInfo(string?, string?)

Formats Info with first line as green caller name and second as message.

Declaration

```
public void DisplayInfo(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DisplayMessage(string?)

Writes the message in the original format.

Declaration

```
public void DisplayMessage(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	String to be printed to console.

DisplayProgressMessage(string?)

Writes message cleanly if progress is running.

Declaration

```
public void DisplayProgressMessage(string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	To display.

DisplayWarning(string?, string?)

Formats Waring using yellow caller name and indented message.

Declaration

```
public void DisplayWarning(string? callerName, string? message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	callerName	Tool or command name.
string	message	String to be printed to console.

DrawProgressAsync(CancellationToken)

Draws progress graphic using arrow stile.

Declaration

```
public Task DrawProgressAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	Used to stop the progress.

Returns

TYPE	DESCRIPTION
Task	

EchoGreeting()

Loads basic info from the resources.

Declaration

```
public void EchoGreeting()
```

EchoHelp()

Loads full help page for the entire fallen-cli from the resources.

Declaration

```
public void EchoHelp()
```

ReadUserInput(string?, bool)

Formats input as a single line with a yellow preface.

Declaration

```
public string? ReadUserInput(string? preface, bool hideInput = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	preface	String that is written before input.

TYPE	NAME	DESCRIPTION
bool	hideInput	If true hides user input.

Returns

TYPE	DESCRIPTION
string	User input.

Implements

ICommandLineFormatter

Namespace FCli.Services.Tools

Classes

AddTool

A tool that validates and adds new commands to storage.

ChangeTool

ConfigTool

GroupTool

IdentityTool

ListTool

A tool that lists all known selectors.

MailTool

PrimesTool

RemoveTool

A tool that removes commands from storage.

RunTool

A tool that runs given command without saving it.

ToolBase

Base class for all known tools.

Class AddTool

A tool that validates and adds new commands to storage.

Inheritance

object
ToolBase
AddTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools
Assembly: FCli.dll

Syntax

```
public class AddTool : ToolBase, ITool, IToolDescriptor
```

Constructors

AddTool()

Empty if used as a descriptor.

Declaration

```
public AddTool()
```

AddTool(ICommandLineFormatter, IResources, IConfig, ICommandLoader, ICommandFactory)

Main constructor.

Declaration

```
public AddTool(ICommandLineFormatter formatter, IResources resources, IConfig config, ICommandLoader
commandLoader, ICommandFactory commandFactory)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
ICommandLoader	commandLoader	
ICommandFactory	commandFactory	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class ChangeTool

Inheritance

object
ToolBase
ChangeTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools

Assembly: FCli.dll

Syntax

```
public class ChangeTool : ToolBase, ITool, IToolDescriptor
```

Constructors

ChangeTool()

Empty if used as a descriptor.

Declaration

```
public ChangeTool()
```

ChangeTool(ICommandLineFormatter, IResources, IConfig, ICommandLoader, ICommandFactory)

Main constructor.

Declaration

```
public ChangeTool(ICommandLineFormatter formatter, IResources resources, IConfig config, ICommandLoader
loader, ICommandFactory factory)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
ICommandLoader	loader	
ICommandFactory	factory	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class ConfigTool

Inheritance

object
ToolBase
ConfigTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools

Assembly: FCli.dll

Syntax

```
public class ConfigTool : ToolBase, ITool, IToolDescriptor
```

Constructors

ConfigTool()

Empty if used as a descriptor.

Declaration

```
public ConfigTool()
```

ConfigTool(ICommandLineFormatter, IResources, IConfig, IEncryptor)

Main constructor.

Declaration

```
public ConfigTool(ICommandLineFormatter formatter, IResources resources, IConfig config, IEncryptor encryptor)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
IEncryptor	encryptor	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class GroupTool

Inheritance

object
ToolBase
GroupTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools

Assembly: FCli.dll

Syntax

```
public class GroupTool : ToolBase, ITool, IToolDescriptor
```

Constructors

GroupTool()

Empty if used as a descriptor.

Declaration

```
public GroupTool()
```

GroupTool(ICommandLineFormatter, IResources, IConfig, ICommandLoader, ICommandFactory)

Main constructor.

Declaration

```
public GroupTool(ICommandLineFormatter formatter, IResources resources, IConfig config, ICommandLoader loader,
ICommandFactory factory)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
ICommandLoader	loader	
ICommandFactory	factory	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class IdentityTool

Inheritance

object
ToolBase
IdentityTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools

Assembly: FCli.dll

Syntax

```
public class IdentityTool : ToolBase, ITool, IToolDescriptor
```

Constructors

IdentityTool()

Empty if used as a descriptor.

Declaration

```
public IdentityTool()
```

IdentityTool(ICommandLineFormatter, IResources, IIdentityManager)

Main constructor.

Declaration

```
public IdentityTool(ICommandLineFormatter formatter, IResources resources, IIdentityManager identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IIdentityManager	identity	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

IToolI
ToolDescriptor

Class ListTool

A tool that lists all known selectors.

Inheritance

object
ToolBase
ListTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools
Assembly: FCli.dll

Syntax

```
public class ListTool : ToolBase, ITool, IToolDescriptor
```

Constructors

ListTool()

Empty if used as a descriptor.

Declaration

```
public ListTool()
```

ListTool(ICommandLineFormatter, IResources, IConfig, ICommandLoader)

Main constructor.

Declaration

```
public ListTool(ICommandLineFormatter formatter, IResources resources, IConfig config, ICommandLoader
commandLoader)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
ICommandLoader	commandLoader	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class MailTool

Inheritance

object
ToolBase
MailTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools
Assembly: FCli.dll

Syntax

```
public class MailTool : ToolBase, ITool, IToolDescriptor
```

Constructors

MailTool()

Empty if used as a descriptor.

Declaration

```
public MailTool()
```

MailTool(ICommandLineFormatter, IResources, IMailer, IIdentityManager)

Main constructor.

Declaration

```
public MailTool(ICommandLineFormatter formatter, IResources resources, IMailer mailer, IIdentityManager
identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IMailer	mailer	
IIdentityManager	identity	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class PrimesTool

Inheritance

object
ToolBase
PrimesTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools

Assembly: FCli.dll

Syntax

```
public class PrimesTool : ToolBase, ITool, IToolDescriptor
```

Constructors

PrimesTool()

Empty if used as a descriptor.

Declaration

```
public PrimesTool()
```

PrimesTool(ICommandLineFormatter, IResources)

Main constructor.

Declaration

```
public PrimesTool(ICommandLineFormatter formatter, IResources resources)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class RemoveTool

A tool that removes commands from storage.

Inheritance

object
ToolBase
RemoveTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools
Assembly: FCli.dll

Syntax

```
public class RemoveTool : ToolBase, ITool, IToolDescriptor
```

Constructors

RemoveTool()

Empty if used as a descriptor.

Declaration

```
public RemoveTool()
```

RemoveTool(ICommandLineFormatter, IResources, ICommandLoader)

Main constructor.

Declaration

```
public RemoveTool(ICommandLineFormatter formatter, IResources resources, ICommandLoader commandLoader)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
ICommandLoader	commandLoader	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class RunTool

A tool that runs given command without saving it.

Inheritance

object
ToolBase
RunTool

Implements

ITool
IToolDescriptor

Inherited Members

ToolBase.Flags
ToolBase.Arg
ToolBase.Formatter
ToolBase.Resources
ToolBase.Execute(string, IEnumerable<Flag>)
ToolBase.FlagHasNoValue(Flag, string)
ToolBase.FlagHasValue(Flag, string)
ToolBase.UnknownFlag(Flag, string)
ToolBase.ValidateUrl(string, string)
ToolBase.ValidatePath(string, string)
ToolBase.ValidateEmail(string, string)
ToolBase.UserConfirm()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools
Assembly: FCli.dll

Syntax

```
public class RunTool : ToolBase, ITool, IToolDescriptor
```

Constructors

RunTool()

Empty if used as a descriptor.

Declaration

```
public RunTool()
```

RunTool(ICommandLineFormatter, IResources, IConfig, ICommandFactory)

Main constructor.

Declaration

```
public RunTool(ICommandLineFormatter formatter, IResources resources, IConfig config, ICommandFactory
commandFactory)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	
IConfig	config	
ICommandFactory	commandFactory	

Properties

Description

Information that should be displayed with `help` flag.

Declaration

```
public override string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Description

Name

Toll's command line selector.

Declaration

```
public override string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ToolBase.Name

Selectors

Known aliases for the selector of the tool.

Declaration

```
public override List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Overrides

ToolBase.Selectors

Type

Unique descriptor for the tool.

Declaration

```
public override ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Overrides

ToolBase.Type

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected override Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Overrides

ToolBase.ActionAsync()

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected override void GuardInit()
```

Overrides

ToolBase.GuardInit()

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected override void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

Overrides

ToolBase.ProcessNextFlag(Flag)

Implements

ITool
IToolDescriptor

Class ToolBase

Base class for all known tools.

Inheritance

object
ToolBase
AddTool
ChangeTool
ConfigTool
GroupTool
IdentityTool
ListTool
MailTool
PrimesTool
RemoveTool
RunTool

Implements

ITool
IToolDescriptor

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: FCli.Services.Tools

Assembly: FCli.dll

Syntax

```
public abstract class ToolBase : ITool, IToolDescriptor
```

Remarks

Contains common properties and some guarding methods.

Constructors

ToolBase()

Default constructor for the descriptors.

Declaration

```
public ToolBase()
```

ToolBase(ICommandLineFormatter, IResources)

Declaration

```
protected ToolBase(ICommandLineFormatter formatter, IResources resources)
```

Parameters

TYPE	NAME	DESCRIPTION
ICommandLineFormatter	formatter	
IResources	resources	

Properties

Arg

Initialized by the Execute method.

Declaration

```
public string Arg { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

Description

Information that should be displayed with `help` flag.

Declaration

```
public abstract string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Flags

List of parsed flags.

Declaration

```
public List<Flag> Flags { get; }
```

Property Value

TYPE	DESCRIPTION
List<Flag>	

Formatter

Pass down to the actual tools.

Declaration

```
protected ICommandLineFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ICommandLineFormatter	

Name

Toll's command line selector.

Declaration

```
public abstract string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Resources

Pass down to the actual tools.

Declaration

```
protected IResources Resources { get; }
```

Property Value

TYPE	DESCRIPTION
IResources	

Selectors

Known aliases for the selector of the tool.

Declaration

```
public abstract List<string> Selectors { get; }
```

Property Value

TYPE	DESCRIPTION
List<string>	

Type

Unique descriptor for the tool.

Declaration

```
public abstract ToolType Type { get; }
```

Property Value

TYPE	DESCRIPTION
ToolType	

Methods

ActionAsync()

Main tool logic, performed after all flags were processed.

Declaration

```
protected abstract Task ActionAsync()
```

Returns

TYPE	DESCRIPTION
Task	

Execute(string, IEnumerable<Flag>)

Performs tool's general logic of processing flags and acting.

Declaration

```
public void Execute(string arg, IEnumerable<Flag> flags)
```

Parameters

TYPE	NAME	DESCRIPTION
string	arg	Tool's arg.
IEnumerable<Flag>	flags	Tool's flags.

FlagHasNoValue(Flag, string)

Asserts that given flag has no value.

Declaration

```
protected void FlagHasNoValue(Flag flag, string toolName)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Flag to test.
string	toolName	

Exceptions

TYPE	CONDITION
FlagException	If value is present.

FlagHasValue(Flag, string)

Asserts that given flag has a value.

Declaration

```
protected void FlagHasValue(Flag flag, string toolName)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Flag to test.
string	toolName	

Exceptions

TYPE	CONDITION
FlagException	If value is missing.

GuardInit()

Performs necessary general validation over received arg and flags.

Declaration

```
protected abstract void GuardInit()
```

Remarks

Can initialize some private values.

ProcessNextFlag(Flag)

Receives each flag sequentially and need to process them accordingly.

Declaration

```
protected abstract void ProcessNextFlag(Flag flag)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Next flag to be processed.

UnknownFlag(Flag, string)

Throws cause given flag is not known to the tool.

Declaration

```
protected void UnknownFlag(Flag flag, string toolName)
```

Parameters

TYPE	NAME	DESCRIPTION
Flag	flag	Unknown flag.
string	toolName	Tool name that doesn't recognize the flag.

Exceptions

TYPE	CONDITION
FlagException	Flag is unknown.

UserConfirm()

Generic user confirmation for an action.

Declaration

```
protected bool UserConfirm()
```

Returns

TYPE	DESCRIPTION
bool	True if confirmed.

ValidateEmail(string, string)

Validates the format of the given email string.

Declaration

```
protected string ValidateEmail(string email, string toolName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	email	Email to check.
string	toolName	Sender.

Returns

TYPE	DESCRIPTION
string	Parsed email string.

Exceptions

TYPE	CONDITION
ArgumentException	If assertion fails.

ValidatePath(string, string)

Asserts that the given path valid and exists.

Declaration

```
protected string ValidatePath(string path, string toolName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	Path to test.
string	toolName	

Returns

TYPE	DESCRIPTION
string	Full path.

Exceptions

TYPE	CONDITION
ArgumentException	If path doesn't exist.

ValidateUrl(string, string)

Asserts that the given URL is valid.

Declaration

```
protected Uri ValidateUrl(string url, string toolName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	url	URL to test.
string	toolName	

Returns

TYPE	DESCRIPTION
Uri	Constructed URI object.

Exceptions

TYPE	CONDITION
ArgumentException	If URI construction fails.

Implements

ITool
IToolDescriptor