System Analysis Report

# Entities

## InstitutionEditModel

|  |  |
| --- | --- |
| Property | Type |
| Id | int |
| Name | string |
| LicenseNumber | string |
| LicenseType | string |
| LicenseIssueDate | System.DateTime |
| LicenseExpiryDate | System.DateTime |
| CountryId | int |
| Address | string |
| IsActive | bool |

## LanguageEditModel

|  |  |
| --- | --- |
| Property | Type |
| Id | int |
| Name | string |
| LanguageCulture | string |
| UniqueSeoCode | string |
| FlagImageFileName | string |
| Rtl | bool |
| Published | bool |
| DisplayOrder | int |
| MakeDefault | bool |

## RegistrationEditModel

|  |  |
| --- | --- |
| Property | Type |
| Id | int |
| InstitutionName | string |
| LicenseNumber | string |
| LicenseType | string |
| LicenseSector | string |
| FinancialType | string |
| IssueDate | System.DateTime |
| ExpiryDate | System.DateTime |
| CountryId | int |
| Address | string |
| Status | string |

## BaseEntity

|  |  |
| --- | --- |
| Property | Type |
| Id | int |

## AuditTrail

|  |  |
| --- | --- |
| Property | Type |
| EntityName | string |
| EntityId | int |
| FieldName | string |
| OldValue | string |
| NewValue | string |
| ChangedByUserId | int |
| ChangedByUser | App.Core.Domain.Users.User |
| ChangedOnUtc | System.DateTime |
| ActionId | int |
| Action | App.Core.Domain.Audit.AuditActionType |
| ClientIp | string |
| Comment | string |

## AuditTrailField

|  |  |
| --- | --- |
| Property | Type |
| AuditTrailTableId | int |
| DBFieldName | string |
| SystemName | string |

## AuditTrailTable

|  |  |
| --- | --- |
| Property | Type |
| DBName | string |
| SystemName | string |

## Attachment

|  |  |
| --- | --- |
| Property | Type |
| FileName | string |
| StoragePath | string |
| MimeType | string |
| Extension | string |
| FileSize | long |
| UploadedByUserId | int |
| UploadedOnUtc | System.DateTime |

## GenericAttribute

|  |  |
| --- | --- |
| Property | Type |
| EntityId | int |
| KeyGroup | string |
| Key | string |
| Value | string |
| StoreId | int |
| CreatedOrUpdatedDateUTC | System.DateTime? |

## Template

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| TemplateType | string |
| Content | string |
| CreatedByUserId | int |
| CreatedOnUtc | System.DateTime |

## Correspondence

|  |  |
| --- | --- |
| Property | Type |
| SenderUserId | int |
| RecipientUserId | int |
| Subject | string |
| Message | string |
| SentOnUtc | System.DateTime |
| Status | string |

## CalendarEvent

|  |  |
| --- | --- |
| Property | Type |
| Title | string |
| Description | string |
| StartDateUtc | System.DateTime |
| EndDateUtc | System.DateTime |
| IsHoliday | bool |
| EventType | string |
| CreatedByUserId | int |

## Country

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| AllowsBilling | bool |
| AllowsShipping | bool |
| TwoLetterIsoCode | string |
| ThreeLetterIsoCode | string |
| NumericIsoCode | int |
| SubjectToVat | bool |
| Published | bool |
| DisplayOrder | int |

## StateProvince

|  |  |
| --- | --- |
| Property | Type |
| CountryId | int |
| Name | string |
| Abbreviation | string |
| Published | bool |
| DisplayOrder | int |

## WorkingLocation

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| Address | string |
| CountryId | int |

## Contract

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| Code | string |
| IsActive | bool |

## Department

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| ParentId | int? |
| IsActive | bool |
| ManagerId | int? |

## Duty

|  |  |
| --- | --- |
| Property | Type |
| EmployeeId | int |
| DutyType | string |
| FromDate | System.DateTime |
| ToDate | System.DateTime |
| Notes | string |

## Institution

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| LicenseNumber | string |
| LicenseTypeId | int |
| LicenseType | App.Core.Domain.Ref.LicenseType |
| LicenseSectorId | int |
| LicenseSector | App.Core.Domain.Ref.LicenseSector |
| FinancialDomainId | int |
| FinancialDomain | App.Core.Domain.Ref.FinancialDomain |
| LicenseIssueDate | System.DateTime |
| LicenseExpiryDate | System.DateTime |
| CountryId | int |
| Address | string |
| IsActive | bool |
| CreatedOnUtc | System.DateTime |
| UpdatedOnUtc | System.DateTime? |
| Departments | System.Collections.Generic.ICollection<App.Core.Domain.Institutions.Department> |
| Jobs | System.Collections.Generic.ICollection<App.Core.Domain.Institutions.Job> |
| Contracts | System.Collections.Generic.ICollection<App.Core.Domain.Institutions.Contract> |
| Duties | System.Collections.Generic.ICollection<App.Core.Domain.Institutions.Duty> |

## Job

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| IsActive | bool |

## Language

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| LanguageCulture | string |
| UniqueSeoCode | string |
| FlagImageFileName | string |
| Rtl | bool |
| Published | bool |
| DisplayOrder | int |

## LocaleStringResource

|  |  |
| --- | --- |
| Property | Type |
| LanguageId | int |
| ResourceName | string |
| ResourceValue | string? |

## LocalizationSettings

|  |  |
| --- | --- |
| Property | Type |
| DefaultAdminLanguageId | int |
| UseImagesForLanguageSelection | bool |
| SeoFriendlyUrlsForLanguagesEnabled | bool |
| AutomaticallyDetectLanguage | bool |
| LoadAllLocaleRecordsOnStartup | bool |
| LoadAllLocalizedPropertiesOnStartup | bool |
| LoadAllUrlRecordsOnStartup | bool |
| IgnoreRtlPropertyForAdminArea | bool |

## LocalizedProperty

|  |  |
| --- | --- |
| Property | Type |
| EntityId | int |
| LanguageId | int |
| LocaleKeyGroup | string |
| LocaleKey | string |
| LocaleValue | string |

## Notification

|  |  |
| --- | --- |
| Property | Type |
| RegistrationId | int? |
| Registration | App.Core.Domain.Registrations.Registration |
| EventTypeId | int |
| EventType | App.Core.Domain.Notifications.NotificationEvent |
| RecipientUserId | int |
| RecipientUser | App.Core.Domain.Users.User |
| TriggeredByUserId | int |
| TriggeredByUser | App.Core.Domain.Users.User |
| Message | string |
| ChannelId | int |
| Channel | App.Core.Domain.Notifications.NotificationChannel |
| StatusId | int |
| Status | App.Core.Domain.Notifications.NotificationDeliveryStatus |
| CreatedOnUtc | System.DateTime |

## NotificationLog

|  |  |
| --- | --- |
| Property | Type |
| NotificationId | int |
| Channel | string |
| Success | bool |
| Response | string |
| SentOnUtc | System.DateTime |

## NotificationReadLog

|  |  |
| --- | --- |
| Property | Type |
| NotificationId | int |
| UserId | int |
| ReadOnUtc | System.DateTime |

## BusinessScaleRange

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| MinValue | int |
| MaxValue | int |
| RangeLabel | string |

## EmployeeRange

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| MinValue | int |
| MaxValue | int |

## FIContact

|  |  |
| --- | --- |
| Property | Type |
| RegistrationId | int |
| ContactTypeId | int |
| ContactType | App.Core.Domain.Registrations.ContactType |
| JobTitle | string |
| FirstName | string |
| MiddleName | string |
| LastName | string |
| CivilId | string |
| PassportId | string |
| CivilAttachmentId | int |
| PassportAttachmentId | int |
| ContactPhone | string |
| BusinessPhone | string |
| Email | string |
| NationalityCountryId | int |
| CreatedOnUtc | System.DateTime |
| UpdatedOnUtc | System.DateTime? |

## FIDocument

|  |  |
| --- | --- |
| Property | Type |
| RegistrationId | int |
| DocumentType | string |
| FilePath | string |
| UploadedByUserId | int |
| UploadedOnUtc | System.DateTime |

## FIRegistrationStatusLog

|  |  |
| --- | --- |
| Property | Type |
| RegistrationId | int |
| RegistrationStatus | App.Core.Domain.Registrations.RegistrationStatus |
| ValidationStatus | App.Core.Domain.Registrations.ValidationStatus? |
| ApprovalStatus | App.Core.Domain.Registrations.ApprovalStatus? |
| AuditStatus | App.Core.Domain.Registrations.AuditStatus? |
| PerformedBy | int |
| ActionDateUtc | System.DateTime |
| Remarks | string |

## Registration

|  |  |
| --- | --- |
| Property | Type |
| InstitutionId | int |
| Institution | App.Core.Domain.Institutions.Institution |
| InstitutionName | string |
| LicenseNumber | string |
| LicenseSectorId | int |
| LicenseSector | App.Core.Domain.Ref.LicenseSector |
| LicenseTypeId | int |
| LicenseType | App.Core.Domain.Ref.LicenseType |
| FinancialDomainId | int |
| FinancialDomain | App.Core.Domain.Ref.FinancialDomain |
| IssueDate | System.DateTime |
| ExpiryDate | System.DateTime |
| CountryId | int |
| Address | string |
| StatusId | int |
| Status | App.Core.Domain.Registrations.RegistrationStatus |
| CreatedByUserId | int |
| UpdatedByUserId | int? |
| SubmittedToUserId | int? |
| CreatedOnUtc | System.DateTime |
| SubmittedDateUtc | System.DateTime? |
| ApprovedDateUtc | System.DateTime? |
| AuditedDateUtc | System.DateTime? |
| BusinessScaleRangeId | int? |
| EmployeeRangeId | int? |

## WorkflowStep

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| NextStepId | int? |
| RoleAllowedId | int |
| RoleAllowed | App.Core.Domain.Security.Role |

## Dashboard

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| Description | string |

## Report

|  |  |
| --- | --- |
| Property | Type |
| Title | string |
| ReportType | string |
| FilePath | string |
| CreatedOnUtc | System.DateTime |

## Permission

|  |  |
| --- | --- |
| Property | Type |
| SystemName | string |
| Category | string |
| Description | string |
| IsActive | bool |

## Privilege

|  |  |
| --- | --- |
| Property | Type |
| SystemName | string |
| DisplayName | string |
| Description | string |

## PrivilegePermission

|  |  |
| --- | --- |
| Property | Type |
| PrivilegeId | int |
| PermissionId | int |

## Role

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| Description | string |
| IsActive | bool |
| CreatedOnUtc | System.DateTime |
| UpdatedOnUtc | System.DateTime? |
| SystemName | string |

## RolePermission

|  |  |
| --- | --- |
| Property | Type |
| RoleId | int |
| PermissionId | int |

## RolePrivilege

|  |  |
| --- | --- |
| Property | Type |
| RoleId | int |
| PrivilegeId | int |

## UserPermissionOverride

|  |  |
| --- | --- |
| Property | Type |
| UserId | int |
| PermissionId | int |
| IsGranted | bool |

## Setting

|  |  |
| --- | --- |
| Property | Type |
| Name | string |
| Value | string |
| StoreId | int |

## User

|  |  |
| --- | --- |
| Property | Type |
| Username | string |
| Email | string |
| PasswordHash | string |
| PasswordSalt | string |
| PasswordFormatId | int |
| PasswordFormat | App.Core.Domain.Users.PasswordFormat |
| IsActive | bool |
| IsLockedOut | bool |
| FailedLoginAttempts | int |
| LockoutEndUtc | System.DateTime? |
| CreatedOnUtc | System.DateTime |
| UpdatedOnUtc | System.DateTime? |
| LastLoginDateUtc | System.DateTime? |
| DeactivatedOnUtc | System.DateTime? |
| InstitutionId | int? |
| RegistrationId | int? |
| PasswordResetToken | string |
| PasswordResetTokenExpiry | System.DateTime? |
| TwoFactorSecret | string |
| TwoFactorEnabled | bool |

## UserLog

|  |  |
| --- | --- |
| Property | Type |
| UserId | int |
| Action | string |
| CreatedOnUtc | System.DateTime |
| ClientIp | string |

## UserPreference

|  |  |
| --- | --- |
| Property | Type |
| UserId | int |
| Language | string |
| LanguageId | int? |
| EnableMfa | bool |
| NotifyByEmail | bool |
| NotifyBySms | bool |
| NotifyInApp | bool |
| UpdatedOnUtc | System.DateTime |

## UserRole

|  |  |
| --- | --- |
| Property | Type |
| UserId | int |
| RoleId | int |
| CreatedOnUtc | System.DateTime |
| UpdatedOnUtc | System.DateTime? |

|  |  |
| --- | --- |
| Property | Type |
| Id | int |
| Hash | string |

## MigrationVersionInfo

|  |  |
| --- | --- |
| Property | Type |
| Version | long |
| Description | string |
| AppliedOn | System.DateTime |
| ApplicationContext | object |
| OwnsSchema | bool |
| SchemaName | string |
| TableName | string |
| ColumnName | string |
| DescriptionColumnName | string |
| UniqueIndexName | string |
| AppliedOnColumnName | string |
| CreateWithPrimaryKey | bool |

# Enums

## DataProviderType

- Unknown

- SqlServer

- MySql

- PostgreSQL

## AuditActionType

- Create

- Update

- Delete

## NotificationChannel

- InApp

- Email

- SMS

- Push

## NotificationDeliveryStatus

- Pending

- Sent

- Failed

## NotificationEvent

- RegistrationSubmitted

- RegistrationValidated

- RegistrationApproved

- RegistrationRejected

- RegistrationReturnedForEdit

- RegistrationAudited

- RegistrationFinalSubmission

- RegistrationArchived

- NewAssignment

## FinancialDomain

- Islamic

- Commercial

## LicenseSector

- Banking

- Exchange

## LicenseType

- Islamic

- Commercial

- Banking

- Exchange

- FinTech

## ApprovalStatus

- Accepted

- Return

- Rejected

## AuditStatus

- Accepted

- Return

- Rejected

## ContactType

- Primary

- Authorized

- Delegate

- BusinessOwner

## RegistrationAction

- Submit

- Validate

- Approve

- Reject

- ReturnForEdit

- Audit

- Archive

- FinalSubmission

## RegistrationStatus

- Draft

- Submitted

- UnderReview

- Approved

- Rejected

- ReturnedForEdit

- Archived

- FinalSubmission

## ValidationStatus

- Draft

- SubmitForApproval

- Accepted

- Return

- Rejected

## PasswordFormat

- Clear

- Hashed

- Encrypted

## DistributedCacheType

- Memory

- SqlServer

- Redis

- RedisSynchronizedMemory

## Lifetime

- Singleton

- Scoped

- Transient

## ServiceLifetime

- Singleton

- Scoped

- Transient

## MigrationProcessType

- NoMatter

- Installation

- Update

## UpdateMigrationType

- Data

- Localization

- Settings

## CrudAction

- Create

- Read

- Update

- Delete

# Services & Interfaces

## ICacheKeyService

App.Core.Caching.CacheKey PrepareKey(App.Core.Caching.CacheKey cacheKey, object[] cacheKeyParameters)

App.Core.Caching.CacheKey PrepareKeyForDefaultCache(App.Core.Caching.CacheKey cacheKey, object[] cacheKeyParameters)

## IAuditTrailService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogAsync(App.Core.Domain.Audit.AuditTrail entry)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogCreateAsync(string entityName, int entityId, int userId, string comment)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogUpdateAsync(string entityName, int entityId, int userId, string field, string oldValue, string newValue, string comment)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogDeleteAsync(string entityName, int entityId, int userId, string comment)

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Audit.AuditTrail>> SearchAsync(System.DateTime? fromUtc, System.DateTime? toUtc, int? changedByUserId, string entityName, int? entityId, string fieldName, App.Core.Domain.Audit.AuditActionType? action, int pageIndex, int pageSize)

## ICalendarService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.CalendarEvent>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Directory.CalendarEvent>>> GetRangeAsync(System.DateTime fromUtc, System.DateTime toUtc)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.CalendarEvent>> CreateAsync(App.Core.Domain.Directory.CalendarEvent model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdateAsync(int id, App.Core.Domain.Directory.CalendarEvent model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## ILanguageService

System.Threading.Tasks.Task<App.Core.Domain.Localization.Language> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Localization.Language>> GetAllAsync(bool onlyPublished)

System.Threading.Tasks.Task InsertAsync(App.Core.Domain.Localization.Language language)

System.Threading.Tasks.Task UpdateAsync(App.Core.Domain.Localization.Language language)

System.Threading.Tasks.Task DeleteAsync(App.Core.Domain.Localization.Language language)

System.Threading.Tasks.Task<App.Core.Domain.Localization.Language> GetCurrentLanguageAsync()

System.Threading.Tasks.Task SetCurrentLanguageAsync(int languageId)

## ILocalizationService

System.Threading.Tasks.Task<string> GetResourceAsync(string key, string defaultValue)

System.Threading.Tasks.Task<App.Core.Domain.Localization.LocaleStringResource> GetByIdAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Localization.LocaleStringResource> GetByNameAsync(string resourceName, int languageId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Localization.LocaleStringResource>> GetAllResourcesAsync(int languageId)

System.Threading.Tasks.Task InsertResourceAsync(App.Core.Domain.Localization.LocaleStringResource resource)

System.Threading.Tasks.Task UpdateResourceAsync(App.Core.Domain.Localization.LocaleStringResource resource)

System.Threading.Tasks.Task DeleteResourceAsync(App.Core.Domain.Localization.LocaleStringResource resource)

System.Threading.Tasks.Task<string> GetLocalizedEnumAsync(TEnum enumValue)

string FormatMessage(string template, object[] args)

## ICorrespondenceService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Correspondences.Correspondence>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Correspondences.Correspondence>>> InboxAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Correspondences.Correspondence>>> OutboxAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Correspondences.Correspondence>> SendAsync(App.Core.Domain.Correspondences.Correspondence model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> MarkAsAsync(int id, string status)

## ICountryService

System.Threading.Tasks.Task<App.Core.Domain.Directory.Country> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Directory.Country>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.Country>> InsertAsync(App.Core.Domain.Directory.Country country)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.Country>> UpdateAsync(App.Core.Domain.Directory.Country country)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Directory.Country> GetByCodeAsync(string twoLetterIsoCode)

System.Threading.Tasks.Task<App.Core.Domain.Directory.Country> GetByNameAsync(string name)

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Directory.Country>> SearchAsync(string keyword, bool? isActive, int pageIndex, int pageSize)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Directory.StateProvince>> GetStatesByCountryIdAsync(int countryId)

System.Threading.Tasks.Task<bool> ExistsAsync(string twoLetterIsoCode)

## IStateProvinceService

System.Threading.Tasks.Task<App.Core.Domain.Directory.StateProvince> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Directory.StateProvince>> GetByCountryIdAsync(int countryId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.StateProvince>> InsertAsync(App.Core.Domain.Directory.StateProvince state)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.StateProvince>> UpdateAsync(App.Core.Domain.Directory.StateProvince state)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## IFileValidationService

System.Threading.Tasks.Task<bool> ValidateAsync(string fileName, long contentLengthBytes)

void Configure(long maxSizeBytes, string[] allowedExtensions)

## IGenericAttributeService

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Common.GenericAttribute>> GetAttributesForEntityAsync(int entityId, string entityName)

System.Threading.Tasks.Task<App.Core.Domain.Common.GenericAttribute> GetAttributeAsync(int entityId, string entityName, string key)

System.Threading.Tasks.Task SetAttributeAsync(int entityId, string entityName, string key, T value)

System.Threading.Tasks.Task DeleteAttributeAsync(int entityId, string entityName, string key)

## IInstallationService

System.Threading.Tasks.Task InstallRequiredDataAsync()

System.Threading.Tasks.Task InstallSampleDataAsync()

## IInstitutionService

System.Threading.Tasks.Task<App.Core.Domain.Institutions.Institution> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Institutions.Institution>> GetAllAsync()

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Institutions.Institution>> InsertAsync(App.Core.Domain.Institutions.Institution institution)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Institutions.Institution>> UpdateAsync(App.Core.Domain.Institutions.Institution institution)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<bool> ExistsAsync(string institutionName, string licenseNumber)

System.Threading.Tasks.Task<bool> IsValidForRegistrationAsync(int institutionId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.Registration>> GetRegistrationsAsync(int institutionId)

System.Threading.Tasks.Task<App.Core.Domain.Registrations.Registration> GetLatestRegistrationAsync(int institutionId)

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Institutions.Institution>> SearchAsync(string name, string licenseNumber, int? countryId, App.Core.Domain.Ref.LicenseSector? licenseSector, App.Core.Domain.Ref.FinancialDomain? financialDomain, int pageIndex, int pageSize)

System.Threading.Tasks.Task<System.Collections.Generic.IDictionary<App.Core.Domain.Ref.LicenseSector, int>> GetInstitutionCountBySectorAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IDictionary<int, int>> GetInstitutionCountByCountryAsync()

## ILookupService

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Ref.BusinessScaleRange>> GetBusinessScaleRangesAsync()

System.Threading.Tasks.Task<App.Core.Domain.Ref.BusinessScaleRange> GetBusinessScaleRangeByIdAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Ref.BusinessScaleRange> FindBusinessScaleRangeAsync(int employeesCount)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Ref.EmployeeRange>> GetEmployeeRangesAsync()

System.Threading.Tasks.Task<App.Core.Domain.Ref.EmployeeRange> GetEmployeeRangeByIdAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Ref.EmployeeRange> FindEmployeeRangeAsync(int employeesCount)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetLicenseTypesAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetLicenseSectorsAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetFinancialDomainsAsync()

System.Threading.Tasks.Task ClearLookupCacheAsync()

## INotificationService

System.Threading.Tasks.Task SendAsync(int? registrationId, App.Core.Domain.Notifications.NotificationEvent eventType, int triggeredByUserId, int recipientUserId, App.Core.Domain.Notifications.NotificationChannel channel, System.Collections.Generic.IDictionary<string, string> tokens)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Notifications.NotificationLog>> GetLogsByUserAsync(int userId)

System.Threading.Tasks.Task MarkAsReadAsync(int notificationId, int userId)

System.Threading.Tasks.Task RetryFailedAsync(int maxAttempts)

## IContactService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.FIContact> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIContact>> GetByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIContact>> InsertAsync(App.Core.Domain.Registrations.FIContact contact)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIContact>> UpdateAsync(App.Core.Domain.Registrations.FIContact contact)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## IDocumentService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.FIDocument> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIDocument>> GetByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIDocument>> InsertAsync(App.Core.Domain.Registrations.FIDocument document)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIDocument>> UpdateAsync(App.Core.Domain.Registrations.FIDocument document)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## IRegistrationActionHintsService

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.RegistrationAction>> GetAvailableActionsAsync(App.Core.Domain.Registrations.Registration registration, int currentUserId, string currentUserRole)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.RegistrationStatus>> GetNextStatusesAsync(App.Core.Domain.Registrations.RegistrationStatus currentStatus, string currentUserRole)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IList<string>>> GetValidationHintsAsync(int registrationId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<int>> GetNotificationRecipientsAsync(App.Core.Domain.Registrations.Registration registration, App.Core.Domain.Registrations.RegistrationAction action, int performedByUserId)

## IRegistrationSearchService

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Registrations.Registration>> SearchAsync(string institutionName, string licenseNumber, int? countryId, App.Core.Domain.Registrations.RegistrationStatus? status, App.Core.Domain.Ref.LicenseType? licenseType, App.Core.Domain.Ref.LicenseSector? licenseSector, App.Core.Domain.Ref.FinancialDomain? financialDomain, System.DateTime? createdFromUtc, System.DateTime? createdToUtc, int pageIndex, int pageSize)

## IRegistrationService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.Registration> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.Registration>> GetAllAsync()

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.Registration>> InsertAsync(App.Core.Domain.Registrations.Registration registration)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.Registration>> UpdateAsync(App.Core.Domain.Registrations.Registration registration)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> SubmitAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ValidateAsync(int registrationId, int performedByUserId, App.Core.Domain.Registrations.ValidationStatus status, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ApproveAsync(int registrationId, int performedByUserId, App.Core.Domain.Registrations.ApprovalStatus status, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AuditAsync(int registrationId, int performedByUserId, App.Core.Domain.Registrations.AuditStatus status, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ReturnForEditAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RejectAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ArchiveAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> FinalSubmissionAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIContact>> GetContactsByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIDocument>> GetDocumentsByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddContactAsync(int registrationId, App.Core.Domain.Registrations.FIContact contact)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddDocumentAsync(int registrationId, App.Core.Domain.Registrations.FIDocument document)

System.Threading.Tasks.Task<bool> CheckDuplicateAsync(string institutionName, string licenseNumber)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIRegistrationStatusLog>> GetStatusHistoryAsync(int registrationId)

System.Threading.Tasks.Task NotifyAsync(int registrationId, App.Core.Domain.Notifications.NotificationEvent eventType, int triggeredByUserId, int recipientUserId, App.Core.Domain.Notifications.NotificationChannel channel, string customMessage)

## IRegistrationStatusLogService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.FIRegistrationStatusLog> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIRegistrationStatusLog>> GetByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIRegistrationStatusLog>> InsertAsync(App.Core.Domain.Registrations.FIRegistrationStatusLog log)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## IRegistrationWorkflowService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> SubmitAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ValidateAsync(int registrationId, int performedByUserId, int recipientUserId, App.Core.Domain.Registrations.ValidationStatus status)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ApproveAsync(int registrationId, int performedByUserId, int recipientUserId, App.Core.Domain.Registrations.ApprovalStatus status)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AuditAsync(int registrationId, int performedByUserId, int recipientUserId, App.Core.Domain.Registrations.AuditStatus status)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ReturnForEditAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RejectAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ArchiveAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> FinalSubmissionAsync(int registrationId, int performedByUserId, int recipientUserId)

## IReportService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Reports.Report>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Reports.Report>>> GetAllAsync(string type)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Reports.Report>> CreateAsync(App.Core.Domain.Reports.Report model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdateAsync(int id, App.Core.Domain.Reports.Report model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## IAccessControlService

System.Threading.Tasks.Task<bool> CanAsync(int userId, string entity, App.Services.Security.CrudAction action)

System.Threading.Tasks.Task<bool> CanDoAsync(int userId, string permissionSystemName)

## IAuthenticationService

System.Threading.Tasks.Task<App.Core.Domain.Users.User> ValidateUserAsync(string username, string password)

System.Threading.Tasks.Task<string> GenerateTokenAsync(App.Core.Domain.Users.User user)

System.Threading.Tasks.Task LogoutAsync(int userId)

System.Threading.Tasks.Task<string> RefreshTokenAsync(string token)

## IEncryptionService

string CreatePasswordHash(string password, string saltKey, App.Core.Domain.Users.PasswordFormat format)

string CreateSaltKey(int size)

bool VerifyPassword(string enteredPassword, string storedHash, string storedSalt, App.Core.Domain.Users.PasswordFormat format)

string GenerateSecureToken(int size)

string EncryptText(string plainText)

string DecryptText(string cipherText)

string ComputeSha256(string input)

string ComputeHmac(string input, string key)

## IPermissionService

System.Threading.Tasks.Task<App.Core.Domain.Security.Permission> GetBySystemNameAsync(string systemName)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Permission>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Permission>> InsertAsync(App.Core.Domain.Security.Permission permission)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Permission>> UpdateAsync(App.Core.Domain.Security.Permission permission)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int permissionId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetPermissionsForRoleAsync(int roleId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> GrantPermissionToRoleAsync(int roleId, string permissionSystemName)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RevokePermissionFromRoleAsync(int roleId, string permissionSystemName)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetUserOverridesAsync(int userId, bool onlyGranted)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> SetUserOverrideAsync(int userId, string permissionSystemName, bool isGranted)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RemoveUserOverrideAsync(int userId, string permissionSystemName)

System.Threading.Tasks.Task<bool> AuthorizeAsync(int userId, string permissionSystemName)

## IRoleService

System.Threading.Tasks.Task<App.Core.Domain.Security.Role> GetByIdAsync(int roleId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Role>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Role>> InsertAsync(App.Core.Domain.Security.Role role)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Role>> UpdateAsync(App.Core.Domain.Security.Role role)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int roleId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Role>> GetRolesByUserIdAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddUserToRoleAsync(int userId, int roleId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RemoveUserFromRoleAsync(int userId, int roleId)

## ISettingService

System.Threading.Tasks.Task<string> GetAsync(string key, string defaultValue)

System.Threading.Tasks.Task<T> GetAsync(string key, T defaultValue)

System.Threading.Tasks.Task SetAsync(string key, string value)

System.Threading.Tasks.Task SetAsync(string key, T value)

System.Threading.Tasks.Task<System.Collections.Generic.IDictionary<string, string>> GetAllAsync(string prefix)

System.Threading.Tasks.Task<bool> HasKeyAsync(string key)

## ITemplateService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Common.Template>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Common.Template>>> GetAllAsync(string type)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Common.Template>> CreateAsync(App.Core.Domain.Common.Template model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdateAsync(int id, App.Core.Domain.Common.Template model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## IUserService

System.Threading.Tasks.Task<App.Core.Domain.Users.User> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Users.User>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Users.User>> InsertAsync(App.Core.Domain.Users.User user, string password, int createdByUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Users.User>> UpdateAsync(App.Core.Domain.Users.User user)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Users.User> GetByEmailAsync(string email)

System.Threading.Tasks.Task<App.Core.Domain.Users.User> GetByUsernameAsync(string username)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ActivateAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeactivateAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ResetPasswordAsync(int userId, string newPassword, int performedByUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LockUserAsync(int userId, int? minutes)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UnlockUserAsync(int userId)

System.Threading.Tasks.Task<bool> CheckPasswordAsync(App.Core.Domain.Users.User user, string password)

System.Threading.Tasks.Task<bool> IsPasswordExpiredAsync(App.Core.Domain.Users.User user)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetRolesAsync(int userId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Role>> GetRoleEntitiesAsync(int userId)

System.Threading.Tasks.Task<bool> IsInRoleAsync(int userId, string roleSystemName)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddToRoleAsync(int userId, int roleId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RemoveFromRoleAsync(int userId, int roleId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Users.User>> GetLockedUsersAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Users.User>> GetInactiveUsersAsync()

System.Threading.Tasks.Task<App.Core.Domain.Users.UserPreference> GetPreferencesAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdatePreferencesAsync(App.Core.Domain.Users.UserPreference preference)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Audit.AuditTrail>> GetUserAuditTrailAsync(int userId, int pageIndex, int pageSize)

## IUserSettingsService

System.Threading.Tasks.Task<App.Core.Domain.Users.UserPreference> GetByUserIdAsync(int userId)

System.Threading.Tasks.Task SetLanguageAsync(int userId, int languageId)

System.Threading.Tasks.Task ToggleMfaAsync(int userId, bool enable)

System.Threading.Tasks.Task UpdateNotificationPreferencesAsync(int userId, bool notifyByEmail, bool notifyBySms, bool notifyInApp)

# Service Implementations

## CacheKeyService

Dependencies: App.Core.AppSettingsConfig.AppSettings

App.Core.Caching.CacheKey PrepareKey(App.Core.Caching.CacheKey cacheKey, object[] cacheKeyParameters)

App.Core.Caching.CacheKey PrepareKeyForDefaultCache(App.Core.Caching.CacheKey cacheKey, object[] cacheKeyParameters)

## AuditTrailService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Audit.AuditTrail>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogAsync(App.Core.Domain.Audit.AuditTrail entry)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogCreateAsync(string entityName, int entityId, int userId, string comment)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogUpdateAsync(string entityName, int entityId, int userId, string field, string oldValue, string newValue, string comment)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LogDeleteAsync(string entityName, int entityId, int userId, string comment)

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Audit.AuditTrail>> SearchAsync(System.DateTime? fromUtc, System.DateTime? toUtc, int? changedByUserId, string entityName, int? entityId, string fieldName, App.Core.Domain.Audit.AuditActionType? action, int pageIndex, int pageSize)

## CalendarService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Directory.CalendarEvent>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.CalendarEvent>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Directory.CalendarEvent>>> GetRangeAsync(System.DateTime fromUtc, System.DateTime toUtc)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.CalendarEvent>> CreateAsync(App.Core.Domain.Directory.CalendarEvent model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdateAsync(int id, App.Core.Domain.Directory.CalendarEvent model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## BaseService

## LanguageService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Localization.Language>

System.Threading.Tasks.Task<App.Core.Domain.Localization.Language> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Localization.Language>> GetAllAsync(bool onlyPublished)

System.Threading.Tasks.Task InsertAsync(App.Core.Domain.Localization.Language language)

System.Threading.Tasks.Task UpdateAsync(App.Core.Domain.Localization.Language language)

System.Threading.Tasks.Task DeleteAsync(App.Core.Domain.Localization.Language language)

System.Threading.Tasks.Task<App.Core.Domain.Localization.Language> GetCurrentLanguageAsync()

System.Threading.Tasks.Task SetCurrentLanguageAsync(int languageId)

## LocalizationService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Localization.LocaleStringResource>, ILanguageService

System.Threading.Tasks.Task<string> GetResourceAsync(string key, string defaultValue)

System.Threading.Tasks.Task<App.Core.Domain.Localization.LocaleStringResource> GetByIdAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Localization.LocaleStringResource> GetByNameAsync(string resourceName, int languageId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Localization.LocaleStringResource>> GetAllResourcesAsync(int languageId)

System.Threading.Tasks.Task InsertResourceAsync(App.Core.Domain.Localization.LocaleStringResource resource)

System.Threading.Tasks.Task UpdateResourceAsync(App.Core.Domain.Localization.LocaleStringResource resource)

System.Threading.Tasks.Task DeleteResourceAsync(App.Core.Domain.Localization.LocaleStringResource resource)

System.Threading.Tasks.Task<string> GetLocalizedEnumAsync(TEnum enumValue)

string FormatMessage(string template, object[] args)

## CorrespondenceService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Correspondences.Correspondence>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Correspondences.Correspondence>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Correspondences.Correspondence>>> InboxAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Correspondences.Correspondence>>> OutboxAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Correspondences.Correspondence>> SendAsync(App.Core.Domain.Correspondences.Correspondence model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> MarkAsAsync(int id, string status)

## CountryService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Directory.Country>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Directory.StateProvince>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Directory.Country> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Directory.Country>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.Country>> InsertAsync(App.Core.Domain.Directory.Country country)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.Country>> UpdateAsync(App.Core.Domain.Directory.Country country)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Directory.Country> GetByCodeAsync(string twoLetterIsoCode)

System.Threading.Tasks.Task<App.Core.Domain.Directory.Country> GetByNameAsync(string name)

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Directory.Country>> SearchAsync(string keyword, bool? isActive, int pageIndex, int pageSize)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Directory.StateProvince>> GetStatesByCountryIdAsync(int countryId)

System.Threading.Tasks.Task<bool> ExistsAsync(string twoLetterIsoCode)

## StateProvinceService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Directory.StateProvince>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Directory.StateProvince> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Directory.StateProvince>> GetByCountryIdAsync(int countryId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.StateProvince>> InsertAsync(App.Core.Domain.Directory.StateProvince state)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Directory.StateProvince>> UpdateAsync(App.Core.Domain.Directory.StateProvince state)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## FileValidationService

void Configure(long maxSizeBytes, string[] allowedExtensions)

System.Threading.Tasks.Task<bool> ValidateAsync(string fileName, long contentLengthBytes)

## GenericAttributeService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Common.GenericAttribute>

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Common.GenericAttribute>> GetAttributesForEntityAsync(int entityId, string entityName)

System.Threading.Tasks.Task<App.Core.Domain.Common.GenericAttribute> GetAttributeAsync(int entityId, string entityName, string key)

System.Threading.Tasks.Task SetAttributeAsync(int entityId, string entityName, string key, T value)

System.Threading.Tasks.Task DeleteAttributeAsync(int entityId, string entityName, string key)

## InstallationService

Dependencies: App.Services.Installation.InstallRequiredData, App.Services.Installation.InstallSampleData

System.Threading.Tasks.Task InstallRequiredDataAsync()

System.Threading.Tasks.Task InstallSampleDataAsync()

## InstitutionService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Institutions.Institution>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.Registration>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Institutions.Institution> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Institutions.Institution>> GetAllAsync()

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Institutions.Institution>> InsertAsync(App.Core.Domain.Institutions.Institution institution)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Institutions.Institution>> UpdateAsync(App.Core.Domain.Institutions.Institution institution)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<bool> ExistsAsync(string institutionName, string licenseNumber)

System.Threading.Tasks.Task<bool> IsValidForRegistrationAsync(int institutionId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.Registration>> GetRegistrationsAsync(int institutionId)

System.Threading.Tasks.Task<App.Core.Domain.Registrations.Registration> GetLatestRegistrationAsync(int institutionId)

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Institutions.Institution>> SearchAsync(string name, string licenseNumber, int? countryId, App.Core.Domain.Ref.LicenseSector? licenseSector, App.Core.Domain.Ref.FinancialDomain? financialDomain, int pageIndex, int pageSize)

System.Threading.Tasks.Task<System.Collections.Generic.IDictionary<App.Core.Domain.Ref.LicenseSector, int>> GetInstitutionCountBySectorAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IDictionary<int, int>> GetInstitutionCountByCountryAsync()

## LookupService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Ref.BusinessScaleRange>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Ref.EmployeeRange>, App.Core.Caching.IStaticCacheManager

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Ref.BusinessScaleRange>> GetBusinessScaleRangesAsync()

System.Threading.Tasks.Task<App.Core.Domain.Ref.BusinessScaleRange> GetBusinessScaleRangeByIdAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Ref.BusinessScaleRange> FindBusinessScaleRangeAsync(int employeesCount)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Ref.EmployeeRange>> GetEmployeeRangesAsync()

System.Threading.Tasks.Task<App.Core.Domain.Ref.EmployeeRange> GetEmployeeRangeByIdAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Ref.EmployeeRange> FindEmployeeRangeAsync(int employeesCount)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetLicenseTypesAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetLicenseSectorsAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetFinancialDomainsAsync()

System.Threading.Tasks.Task ClearLookupCacheAsync()

## NotificationService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Notifications.Notification>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Notifications.NotificationLog>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Notifications.NotificationReadLog>, App.Services.Localization.ILocalizationService, App.Services.Templates.ITemplateService

System.Threading.Tasks.Task SendAsync(int? registrationId, App.Core.Domain.Notifications.NotificationEvent eventType, int triggeredByUserId, int recipientUserId, App.Core.Domain.Notifications.NotificationChannel channel, System.Collections.Generic.IDictionary<string, string> tokens)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Notifications.NotificationLog>> GetLogsByUserAsync(int userId)

System.Threading.Tasks.Task MarkAsReadAsync(int notificationId, int userId)

System.Threading.Tasks.Task RetryFailedAsync(int maxAttempts)

## ContactService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.FIContact>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.FIContact> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIContact>> GetByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIContact>> InsertAsync(App.Core.Domain.Registrations.FIContact contact)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIContact>> UpdateAsync(App.Core.Domain.Registrations.FIContact contact)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## DocumentService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.FIDocument>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.FIDocument> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIDocument>> GetByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIDocument>> InsertAsync(App.Core.Domain.Registrations.FIDocument document)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIDocument>> UpdateAsync(App.Core.Domain.Registrations.FIDocument document)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## RegistrationActionHintsService

Dependencies: App.Services.Registrations.IContactService, App.Services.Registrations.IDocumentService, App.Services.Registrations.IRegistrationStatusLogService

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.RegistrationAction>> GetAvailableActionsAsync(App.Core.Domain.Registrations.Registration registration, int currentUserId, string currentUserRole)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.RegistrationStatus>> GetNextStatusesAsync(App.Core.Domain.Registrations.RegistrationStatus currentStatus, string currentUserRole)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IList<string>>> GetValidationHintsAsync(int registrationId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<int>> GetNotificationRecipientsAsync(App.Core.Domain.Registrations.Registration registration, App.Core.Domain.Registrations.RegistrationAction action, int performedByUserId)

## RegistrationSearchService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.Registration>

System.Threading.Tasks.Task<App.Services.Common.PagedResult<App.Core.Domain.Registrations.Registration>> SearchAsync(string institutionName, string licenseNumber, int? countryId, App.Core.Domain.Registrations.RegistrationStatus? status, App.Core.Domain.Ref.LicenseType? licenseType, App.Core.Domain.Ref.LicenseSector? licenseSector, App.Core.Domain.Ref.FinancialDomain? financialDomain, System.DateTime? createdFromUtc, System.DateTime? createdToUtc, int pageIndex, int pageSize)

## RegistrationService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.Registration>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.FIContact>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.FIDocument>, App.Services.Localization.ILocalizationService, App.Services.Registrations.IRegistrationStatusLogService, App.Services.Notifications.INotificationService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.Registration> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.Registration>> GetAllAsync()

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.Registration>> InsertAsync(App.Core.Domain.Registrations.Registration registration)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.Registration>> UpdateAsync(App.Core.Domain.Registrations.Registration registration)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> SubmitAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ValidateAsync(int registrationId, int performedByUserId, App.Core.Domain.Registrations.ValidationStatus status, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ApproveAsync(int registrationId, int performedByUserId, App.Core.Domain.Registrations.ApprovalStatus status, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AuditAsync(int registrationId, int performedByUserId, App.Core.Domain.Registrations.AuditStatus status, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ReturnForEditAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RejectAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ArchiveAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> FinalSubmissionAsync(int registrationId, int performedByUserId, string remarks)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIContact>> GetContactsByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIDocument>> GetDocumentsByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddContactAsync(int registrationId, App.Core.Domain.Registrations.FIContact contact)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddDocumentAsync(int registrationId, App.Core.Domain.Registrations.FIDocument document)

System.Threading.Tasks.Task<bool> CheckDuplicateAsync(string institutionName, string licenseNumber)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIRegistrationStatusLog>> GetStatusHistoryAsync(int registrationId)

System.Threading.Tasks.Task NotifyAsync(int registrationId, App.Core.Domain.Notifications.NotificationEvent eventType, int triggeredByUserId, int recipientUserId, App.Core.Domain.Notifications.NotificationChannel channel, string customMessage)

## RegistrationStatusLogService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Registrations.FIRegistrationStatusLog>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Registrations.FIRegistrationStatusLog> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Registrations.FIRegistrationStatusLog>> GetByRegistrationIdAsync(int registrationId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Registrations.FIRegistrationStatusLog>> InsertAsync(App.Core.Domain.Registrations.FIRegistrationStatusLog log)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## RegistrationWorkflowService

Dependencies: App.Services.Registrations.IRegistrationService, App.Services.Notifications.INotificationService, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> SubmitAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ValidateAsync(int registrationId, int performedByUserId, int recipientUserId, App.Core.Domain.Registrations.ValidationStatus status)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ApproveAsync(int registrationId, int performedByUserId, int recipientUserId, App.Core.Domain.Registrations.ApprovalStatus status)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AuditAsync(int registrationId, int performedByUserId, int recipientUserId, App.Core.Domain.Registrations.AuditStatus status)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ReturnForEditAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RejectAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ArchiveAsync(int registrationId, int performedByUserId, int recipientUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> FinalSubmissionAsync(int registrationId, int performedByUserId, int recipientUserId)

## ReportService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Reports.Report>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Reports.Report>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Reports.Report>>> GetAllAsync(string type)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Reports.Report>> CreateAsync(App.Core.Domain.Reports.Report model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdateAsync(int id, App.Core.Domain.Reports.Report model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## AccessControlService

Dependencies: App.Services.Security.IPermissionService

System.Threading.Tasks.Task<bool> CanAsync(int userId, string entity, App.Services.Security.CrudAction action)

System.Threading.Tasks.Task<bool> CanDoAsync(int userId, string permissionSystemName)

## AuthenticationService

Dependencies: App.Services.Users.IUserService, App.Services.Security.IEncryptionService, Microsoft.Extensions.Configuration.IConfiguration

System.Threading.Tasks.Task<App.Core.Domain.Users.User> ValidateUserAsync(string username, string password)

System.Threading.Tasks.Task<string> GenerateTokenAsync(App.Core.Domain.Users.User user)

System.Threading.Tasks.Task LogoutAsync(int userId)

System.Threading.Tasks.Task<string> RefreshTokenAsync(string token)

## EncryptionService

Dependencies: Microsoft.Extensions.Options.IOptions<App.Core.Configuration.SecuritySettings>

string CreateSaltKey(int size)

string CreatePasswordHash(string password, string saltKey, App.Core.Domain.Users.PasswordFormat format)

bool VerifyPassword(string enteredPassword, string storedHash, string storedSalt, App.Core.Domain.Users.PasswordFormat format)

string GenerateSecureToken(int size)

string ComputeSha256(string input)

string ComputeHmac(string input, string key)

string EncryptText(string plainText)

string DecryptText(string cipherText)

## PermissionService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Security.Permission>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Security.RolePermission>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Users.UserRole>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Security.UserPermissionOverride>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Security.Permission> GetBySystemNameAsync(string systemName)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Permission>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Permission>> InsertAsync(App.Core.Domain.Security.Permission permission)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Permission>> UpdateAsync(App.Core.Domain.Security.Permission permission)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int permissionId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetPermissionsForRoleAsync(int roleId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> GrantPermissionToRoleAsync(int roleId, string permissionSystemName)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RevokePermissionFromRoleAsync(int roleId, string permissionSystemName)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetUserOverridesAsync(int userId, bool onlyGranted)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> SetUserOverrideAsync(int userId, string permissionSystemName, bool isGranted)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RemoveUserOverrideAsync(int userId, string permissionSystemName)

System.Threading.Tasks.Task<bool> AuthorizeAsync(int userId, string permissionSystemName)

## RoleService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Security.Role>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Users.UserRole>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Security.Role> GetByIdAsync(int roleId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Role>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Role>> InsertAsync(App.Core.Domain.Security.Role role)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Security.Role>> UpdateAsync(App.Core.Domain.Security.Role role)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int roleId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Role>> GetRolesByUserIdAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddUserToRoleAsync(int userId, int roleId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RemoveUserFromRoleAsync(int userId, int roleId)

## SettingService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Settings.Setting>

System.Threading.Tasks.Task<string> GetAsync(string key, string defaultValue)

System.Threading.Tasks.Task<T> GetAsync(string key, T defaultValue)

System.Threading.Tasks.Task SetAsync(string key, string value)

System.Threading.Tasks.Task SetAsync(string key, T value)

System.Threading.Tasks.Task<System.Collections.Generic.IDictionary<string, string>> GetAllAsync(string prefix)

System.Threading.Tasks.Task<bool> HasKeyAsync(string key)

## TemplateService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Common.Template>, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Common.Template>> GetAsync(int id)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<System.Collections.Generic.IReadOnlyList<App.Core.Domain.Common.Template>>> GetAllAsync(string type)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Common.Template>> CreateAsync(App.Core.Domain.Common.Template model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdateAsync(int id, App.Core.Domain.Common.Template model)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

## UserService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Users.User>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Security.Role>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Users.UserRole>, App.Core.RepositoryServices.IRepository<App.Core.Domain.Users.UserPreference>, App.Services.Security.IEncryptionService, App.Services.Audit.IAuditTrailService, App.Services.Localization.ILocalizationService

System.Threading.Tasks.Task<App.Core.Domain.Users.User> GetByIdAsync(int id)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Users.User>> GetAllAsync(bool onlyActive)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Users.User>> InsertAsync(App.Core.Domain.Users.User user, string password, int createdByUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult<App.Core.Domain.Users.User>> UpdateAsync(App.Core.Domain.Users.User user)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeleteAsync(int id)

System.Threading.Tasks.Task<App.Core.Domain.Users.User> GetByEmailAsync(string email)

System.Threading.Tasks.Task<App.Core.Domain.Users.User> GetByUsernameAsync(string username)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ActivateAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> DeactivateAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> ResetPasswordAsync(int userId, string newPassword, int performedByUserId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> LockUserAsync(int userId, int? minutes)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UnlockUserAsync(int userId)

System.Threading.Tasks.Task<bool> CheckPasswordAsync(App.Core.Domain.Users.User user, string password)

System.Threading.Tasks.Task<bool> IsPasswordExpiredAsync(App.Core.Domain.Users.User user)

System.Threading.Tasks.Task<System.Collections.Generic.IList<string>> GetRolesAsync(int userId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Security.Role>> GetRoleEntitiesAsync(int userId)

System.Threading.Tasks.Task<bool> IsInRoleAsync(int userId, string roleSystemName)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> AddToRoleAsync(int userId, int roleId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> RemoveFromRoleAsync(int userId, int roleId)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Users.User>> GetLockedUsersAsync()

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Users.User>> GetInactiveUsersAsync()

System.Threading.Tasks.Task<App.Core.Domain.Users.UserPreference> GetPreferencesAsync(int userId)

System.Threading.Tasks.Task<App.Services.Common.ServiceResult> UpdatePreferencesAsync(App.Core.Domain.Users.UserPreference preference)

System.Threading.Tasks.Task<System.Collections.Generic.IList<App.Core.Domain.Audit.AuditTrail>> GetUserAuditTrailAsync(int userId, int pageIndex, int pageSize)

## UserSettingsService

Dependencies: App.Core.RepositoryServices.IRepository<App.Core.Domain.Users.UserPreference>, ILanguageService

System.Threading.Tasks.Task<App.Core.Domain.Users.UserPreference> GetByUserIdAsync(int userId)

System.Threading.Tasks.Task SetLanguageAsync(int userId, int languageId)

System.Threading.Tasks.Task ToggleMfaAsync(int userId, bool enable)

System.Threading.Tasks.Task UpdateNotificationPreferencesAsync(int userId, bool notifyByEmail, bool notifyBySms, bool notifyInApp)

# Services Dependency Map

