インストールの一般的な詳細

Version 2024.1
2024-06-03

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

インストールの一般的な詳細
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

1 インストール・ディレクトリ 1

2 セットアップ・タイプ 3
- 2.1 セットアップ・タイプ 3
- 2.2 既存の CSP ゲートウェイ 4

3 文字幅設定 7

4 ポート番号 9

5 インストール・マニフェストの作成および使用 11
- 5.1 インストール・マニフェストの概要 11
- 5.2 インストール・マニフェストの作成 11
  - 5.2.1 マニフェスト・クラス定義 11
  - 5.2.2 マニフェストの一般的なオプション 12
  - 5.2.3 〈Manifest〉 タグ内の変数 13
  - 5.2.4 〈Manifest〉 タグのリスト 14
- 5.3 マニフェストの使用 25
  - 5.3.1 手動で実行する場合 26
  - 5.3.2 インストール時に実行する場合 26
- 5.4 インストール・マニフェストの例 27
  - 5.4.1 空のテンプレート 27
  - 5.4.2 ネームスペースの作成 27
  - 5.4.3 アップグレード後のコンパイル 28

6 複数の InterSystems IRIS インスタンスの構成 31

7 InterSystems IRIS の言語の変更 33

8 Web サーバのセットアップのトラブルシューティング 35
- 8.1 ドキュメント・リンクが機能しない 35
- 8.2 Windows と IIS 35
  - 8.2.1 VS Code のトラブルシューティング 35
  - 8.2.2 IIS の再起動 35
  - 8.2.3 インスタンスの修復または変更 36

テーブル一覧

テーブル 2-1: セットアップ・タイプ別のインストールされるコンポーネント 4

1

インストール・ディレクトリ

インストール・ディレクトリは、InterSystems IRIS インスタンスをインストールするディレクトリです。このドキュメントでは、このディレクトリを install-dir と呼びます。InterSystems IRIS のインストール後にインストール・ディレクトリを変更することはできません。

install-dir として指定できるディレクトリには、いくつかの制限があります。ディレクトリは、シンボリック・リンクを含まない、完全に解決された物理パスである必要があります。ディレクトリ名には、US ASCII 文字セットの文字のみを使用でき、キャレット (^) は使用できません。また、InterSystems IRIS を以下のディレクトリにインストールすることはできません。

- UNC (ローカルでない) パスのディレクトリ。
- ドライブのルート・レベル (C:¥ など) にあるディレクトリ。
- ¥Program Files ディレクトリ下の任意の場所にあるディレクトリ。

インストール時に指定しない場合、install-dir には既定値が使用されます。以下の表に示すように、この既定値は、プラットフォーム、インストール・タイプ、およびユーザの選択によって異なります。

プラットフォーム	インストール・タイプ	既定のディレクトリ
Windows	手動	インストール・ユーザが他に指定しない限り、C:¥InterSystems¥Iris (複数インスタンスが存在する場合は、IrisN)。
	自動	INSTALLDIR プロパティで他に指定されない限り、C:¥InterSystems¥Iris (複数インスタンスが存在する場合は、IrisN)。
UNIX®、Linux、macOS	手動	既定値なし。インストール・ユーザが指定する必要があります。/home ディレクトリ、そのサブディレクトリ、または /usr/local/etc/irissys ディレクトリは選択しないでください。
	自動	既定値なし。ISC_PACKAGE_INSTALLDIR パラメータが必要です。

2
セットアップ・タイプ

インストール時に、インストールする InterSystems IRIS コンポーネントを選択できます。オプションは以下のとおりです。

2.1 セットアップ・タイプ

- **[開発]** – InterSystems IRIS データベース・エンジン (User データベース、言語ゲートウェイ、サーバ監視ツール)、Web ゲートウェイ、スタジオ (Windows のみ)、すべてのサポート対象言語バインディング、データベース・ドライバを含めます。このインスタンスをクライアントのタスクとサーバのタスクの両方を実行するために使用する場合は、このオプションを選択します。
- **[サーバ]** – InterSystems IRIS データベース・エンジン (User データベース、言語ゲートウェイ、サーバ監視ツール) および Web ゲートウェイを含めます。InterSystems IRIS クライアントからアクセス可能な InterSystems IRIS データベース・サーバとしてこのインスタンスを使用する場合は、このオプションを選択します。
- **[カスタム]** – インストールまたはアンインストールする特定のコンポーネントを指定できます。特定の InterSystems IRIS コンポーネントをインストールまたは削除することを計画している場合、このオプションを選択します。

Windows では、以下の 2 つの追加セットアップ・タイプから選択できます。

- **[クライアント]** – スタジオ、ODBC ドライバ、JDBC ドライバ、および InterSystems IRIS アプリケーション開発コンポーネントを含めます。このコンピュータまたは別のコンピュータで、InterSystems IRIS データベース・サーバのクライアントとしてこのインスタンスを使用する場合は、このオプションを選択します。
- **[Web サーバ]** – Web ゲートウェイが含まれます。InterSystems IRIS の機能のうち、Web ゲートウェイで必要なもののみをインストールする場合は、このオプションを選択します。

以下のテーブルでは、それぞれのセットアップ・タイプでインストールされるコンポーネントを示しています。カスタム・インストールを実行する場合は、グループから個別のコンポーネントのみを含めることができます。

テーブル 2-1: セットアップ・タイプ別のインストールされるコンポーネント

コンポーネント・グループ	コンポーネント	開発	サーバ	クライアント	Web
InterSystems IRIS データベース・エンジン (InterSystems IRIS サーバ)	サーバ監視ツール User データベース 言語ゲートウェイ エージェント・サービス (ISCAgent) Apache FOP (Formatting Objects Processor) InterSystems IntegratedML	X	X		
InterSystems IRIS ランチャー (Windows のみ)		X	X	X	
スタジオ (Windows のみ)		X		X	
データベース・ドライバ	ODBC ドライバ Java Database Connectivity	X		X	
InterSystems IRIS アプリケーション開発	InterSystems IRIS 用の Java バインディング InterSystems IRIS 共有ライブラリ・サポート InterSystems IRIS 用の .NET バインディング スレッド・サーバ・ライブラリ その他のサンプル	X		X	
Web ゲートウェイ	IIS 用 Web ゲートウェイ・バイナリ Apache2.4.x 用 Web ゲートウェイ・バイナリ	X	X		X

2.2 既存の CSP ゲートウェイ

Web ゲートウェイをインストールする際に、お使いのシステムに CSP ゲートウェイが既にインストールされている場合は、インストーラにより CSP ゲートウェイが Web ゲートウェイに自動的にアップグレードされます。ただし、インストーラは CSP.ini ファイルの新しいコピーを作成します。CSP ゲートウェイの CSP.ini ファイルを保持するには、以下の手順に従います。

1. install-dir/CSP/bin にある CSP.ini ファイルのバックアップを作成します。
2. Web ゲートウェイ・インストーラを実行します。
3. CSP.ini のバックアップをリストアします。

注釈　Web ゲートウェイをインストールしても、古い接続は閉じられません。これらの古い接続を削除するには、[システムステータス] ページで手動で閉じる必要があります。

3

文字幅設定

インストールには、8 ビットまたは Unicode (16 ビット) の文字幅を選択する必要があります。8 ビット・インスタンスは 16 ビット形式のデータを処理できませんが、Unicode インスタンスは 8 ビット・データと 16 ビット・データの両方を処理できます。

インスタンスで、Unicode 文字のみを使用する言語 (日本語など) のデータを格納および処理する必要がある場合は、Unicode を選択してください。インスタンスで使用するロケールに設定されている基本文字セットが、Latin-1 文字セット ISO8859-1 に基づいている場合は、8 ビットを選択できますが、以下の点に留意してください。

- 8 ビットを選択した場合、インスタンスによって格納されるデータは、同じ文字セットに基づく 8 ビット・ロケールにのみ移植できます。
- Unicode を選択した場合、インスタンスによって格納されるデータは 8 ビット・インスタンスに移植できません。

4

ポート番号

標準インストールでは、InterSystems IRIS インスタンスに対して以下のポート番号が設定されます。

- スーパーサーバのポート番号 ー 1972。取得されている場合は、51773 またはこれ以降の使用可能な最初のポート番号。
- Telnet ー 23

新規インストール時に**カスタム・インストール**を実行することで、別のポート番号を割り当てることができます ("セットアップ・タイプ" を参照)。65535 よりも大きなポート番号は入力できません。

既にインストールされている InterSystems IRIS インスタンス上のポート番号の設定に関する詳細は、"ポート番号の設定" を参照してください。

5
インストール・マニフェストの作成および使用

このトピックでは、特定の InterSystems IRIS® Data Platform 構成を記したインストール・マニフェストを作成し、これを使用して InterSystems IRIS インスタンスを構成するコードを生成するための %Installer ユーティリティの使用法について説明します。

5.1 インストール・マニフェストの概要

%Installer ユーティリティによりインストール・マニフェストを定義します。これは段階的なインストール・プロセスではなく、特定の InterSystems IRIS 構成を記述および構成するものです。これには、通常インストール中 (スーパーサーバ・ポート、オペレーティング・システムなど) に指定した情報を含む変数を使用して、目的の構成を記述した XData ブロックを含むクラスを作成します。また、このクラスにインスタンス構成用のコードを生成するための XData ブロックを使用するメソッドを組み込みます。この付録では、インストール・マニフェストの例を提供します。この例をコピーして貼り付けて、作業を開始することができます。

マニフェストを定義すると、インストール中、またはターミナル・セッションやコードからマニフェストを呼び出すことができます。マニフェストは %SYS ネームスペースで実行する必要があります。

5.2 インストール・マニフェストの作成

このセクションでは、インストール・マニフェストの作成に必要な情報を以下の内容に分けて説明します。

- マニフェスト・クラス定義
- マニフェストの一般的なオプション
- 〈Manifest〉 タグ内の変数
- 〈Manifest〉 タグのリスト

5.2.1 マニフェスト・クラス定義

インストール・マニフェストを定義するクラスを作成するには、以下の条件を満たすクラスを作成します (または、以下の "インストール・マニフェストの例" のセクションの空白のテンプレートから開始します)。

- このクラスには、%occInclude インクルード・ファイルを含める必要があります。

- このクラスには、インストールの詳細を指定する XData ブロックを含める必要があります。XData ブロックの名前は、後で引数として使用されます。XData ブロックのルート要素は <Manifest> とする必要があります。詳細は、“〈Manifest〉 タグのリスト” を参照してください。

  スタジオでは、XData ブロックの名前の後に [XMLNamespace = INSTALLER] を追加すると、XData ブロックの入力に応じてサポートが提供されます。

- このクラスは、名前で XData ブロックを参照する setup() メソッドを定義する必要があります (″インストール・マニフェストの例″ のセクションを参照)。

5.2.2 マニフェストの一般的なオプション

このセクションでは、インストール・マニフェストを使用して実行するいくつかの一般的なタスクについて説明します。以下のオプションについて説明します。

- ネームスペースの定義
- ユーザとパスワードの追加
- マニフェスト・ログへの書き込み
- アップグレード後のタスクの実行

5.2.2.1 ネームスペースの定義

ネームスペースを定義するには、<Manifest> タグ内に任意の数の <Namespace> タグを追加します。

ネームスペースのデータベースまたはマッピングを定義するには、<Namespace> タグの中に <Configuration> タグを入れます。ネームスペースに含まれるデータベースごとに、<Configuration> タグ内に <Database> タグを追加します。マッピングを定義するには、該当する <Database> タグの下に、<GlobalMapping> タグ、<RoutineMapping> タグ、および <ClassMapping> タグを追加します。</Configuration> タグは、定義されたマッピングを有効化します。

さらに、<Import> タグを使用して、<Namespace> タグ内にグローバル、ルーチン、およびクラスをロードできます。<Invoke> タグでクラス・メソッドを起動してルーチンを実行し、インポートされているグローバルにアクセスすることもできます。

<Var> タグで変数を定義できます。マニフェスト内の変数を参照するには、${var_name} 構文を使用します。詳細は、以下の ″〈Manifest〉 タグ内の変数″ を参照してください。

5.2.2.2 ユーザとパスワードの追加

インストールしたインスタンスにユーザ (ロールとパスワードを含む) を追加する方法は複数あります。

- ″〈Manifest〉 タグのリスト″ の説明に従って、インストール・マニフェストに <User> タグを含めます。

  <User> タグの PasswordVar パラメータはユーザのパスワードを格納した変数を指定します。例えば、PasswordVar="Pwd" を定義すると、変数 Pwd の値がユーザのパスワードに指定されます。この変数にデータを移入する方法は多様ですが、移入方法は担当者次第です。InterSystems IRIS または Web サービスの別のインスタンスに対してメソッドをリモートで呼び出すことも検討できますが、この方法の問題点は、InterSystems IRIS がインストールされているサーバでインターネット・アクセスが必要な場合があることです。インストールしているインスタンスに対して使用するメソッドをインポートするか、ユーザとパスワード (マニフェストに渡すことができます) の入力を求めるインストールにクライアント側フォームを追加する方法も考えられます。

- インストール完了後に管理ポータルでユーザを編集します (″ユーザ″ を参照)。

- InterSystems IRIS のステージング・インスタンスで Security.Users クラスを使用します (以下を参照)。

1. Security.Users.Export() メソッドを使用して、ユーザ情報をエクスポートします。
2. ユーザ情報をインポートするには、(%SYS ネームスペースの) マニフェスト・クラスの先頭で以下を追加します。

```
<Invoke Class="Security.Users" Method="Import" CheckStatus="true">
<Arg Value="PathToExportedUserInformation"/>
</Invoke>
```

PathToExportedUserInformation は、Security.Users.Export() メソッドで指定された出力ファイルの場所です。

5.2.2.3 マニフェスト・ログへの書き込み

次の形式でクラスに 〈Log〉 タグを組み込むことで、マニフェスト・ログに追加するメッセージを定義できます。

```
<Log Level="<level>" Text="<text>"/>
```

ログのレベルは -1 から 3 までの範囲になる必要があります。-1 は “none” (なし)、3 は “verbose” (詳細) です。setup() メソッドで指定されたログのレベルが <Log> タグの Level プロパティの値以上の場合にメッセージがログに記録されます。テキストは 32,000 文字に制限されます。

setup() メソッドの 2 番目の引数によりレベルを設定します (詳細は、この付録で後述の “マニフェストの使用” を参照)。これはインストールからマニフェストへ渡すことはできないため、次のようにクラスに設定する必要があります。

```
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
```

メッセージの表示位置を setup() メソッドの 4 番目の引数 (%Installer.AbstractLogger) により指定できます。例えば、オペレーティング・システム・ファイルを参照する %Installer.FileLogger オブジェクトをインスタンス化する場合、すべての %Installer およびログ・メッセージはそのファイルに書き込まれます。(この引数がないと、setup() が直接呼び出された場合には、すべてのメッセージがプライマリ・デバイスに書き込まれ、インストーラによって実行された場合には無視されます。)

5.2.2.4 アップグレード後のタスクの実行

InterSystems IRIS のインスタンスをアップグレードする際、インストール・マニフェストを使用して、コードのリコンパイルなど、必要なアップグレード後のタスクを自動的に実行できます。そのためには、<Invoke> タグを使用して、このドキュメントの “InterSystems IRIS のアップグレード” の章の ″ネームスペースのコンパイル方法″ のセクションに記載されているクラス・メソッドを呼び出します。

アップグレード時に使用できるマニフェストの例は、この付録の ″インストール・マニフェストの例″ のセクションを参照してください。

5.2.3 〈Manifest〉 タグ内の変数

一部のタグは、マニフェストが実行されると展開される式 (文字列) を含めることができます。展開できる式には 3 種類あります (下記参照)。

${<var_name>} – 変数

変数の値に展開されます。このセクションで説明する事前定義変数のほかに、〈Var〉 タグで追加の変数を指定できます。

${#<param_name>} – クラス・パラメータ

マニフェスト・クラスの指定されたパラメータの値に展開します。

#{<ObjectScript_expression>} — ObjectScript 式

指定された InterSystems IRIS Object Script 式に展開します (正しく引用符を付ける必要があります)。

注釈　パラメータの式はコンパイル時に展開されます。つまり、変数および ObjectScript 式の内側で入れ子にできます。また、変数の式は ObjectScript 式の前に展開されるため、後者の内側で入れ子にできます。

以下のテーブルに、マニフェストで使用できる事前定義変数を示します。

変数名	説明
SourceDir	(インストーラ実行中にのみ使用可能) インストール (setup_irisdb.exe または irisinstall) の実行元となるディレクトリ。
ISCUpgrade	(インストーラ実行中にのみ使用可能) 新規インストールまたはアップグレードであるかを示します。この変数は 0 (新規インストール) または 1 (アップグレード) のいずれかになります。
CFGDIR	"INSTALLDIR" を参照してください。
CFGNAME	インスタンス名。
CPUCOUNT	オペレーティング・システムの CPU の数。
CSPDIR	CSP ディレクトリ。
HOSTNAME	ホスト・サーバの名前。
INSTALLDIR	InterSystems IRIS がインストールされるディレクトリ。
MGRDIR	管理者用 (mgr) ディレクトリ。
PLATFORM	オペレーティング・システム。
PORT	InterSystems IRIS スーパーサーバ・ポート。
PROCESSOR	プロセッサ・チップ。
VERSION	InterSystems IRIS のバージョン番号。

5.2.4 <Manifest> タグのリスト

コード生成に関するすべての情報は、最も外側の XML タグ (<Manifest>) 内に含まれます。<Manifest> 内の各タグは、InterSystems IRIS の特定の構成を記述します。このセクションでは、これらのタグとその関数のリストを示します。ここでは、すべてのタグと属性は記載されていません。完全なリストについては、%Installer クラスリファレンスのドキュメントを参照してください。属性に既定値がある場合は、各属性の横に角括弧で囲んでリストされています。

重要　NULL 引数はマニフェスト・クラスのタグで渡すことはできません。例えば、<Arg/> タグは空の文字列を渡すので、<Arg=""/> と同等となります。NULL ではありません。

- <Arg>
- <ClassMapping>
- <Compile>
- <Configuration>
- <CopyClass>
- <CopyDir>
- <CopyFile>

- 〈CSPApplication〉
- 〈Credential〉
- 〈Database〉
- 〈Default〉
- 〈Else〉
- 〈Error〉
- 〈ForEach〉
- 〈GlobalMapping〉
- 〈If〉
- 〈IfDef〉
- 〈IfNotDef〉
- 〈Import〉
- 〈Invoke〉
- 〈LoadPage〉
- 〈Log〉
- 〈Manifest〉
- 〈Namespace〉
- 〈Production〉
- 〈Resource〉
- 〈Role〉
- 〈RoutineMapping〉
- 〈Setting〉
- 〈SystemSetting〉
- 〈User〉
- 〈Var〉

<Arg>

親タグ：〈Invoke〉、〈Error〉

<Invoke> または <Error> から呼び出されるメソッドに引数を渡します。

- Value—引数の値。

```
<Error Status="$$$NamespaceDoesNotExist">
        <Arg Value="${NAMESPACE}"/>
/>
```

<ClassMapping>

親タグ：〈Configuration〉

データベースからこの <Configuration> 項目を含むネームスペースへのクラス・マッピングを作成します。

- Package—マップするパッケージ。
- From—マッピングに使用するソース・データベース。

```
<ClassMapping Package="MYAPP"
     From="MYDB"/>
```

<Compile>

親タグ：〈Namespace〉

%SYSTEM.OBJ.Compile(Class, Flags) を呼び出して、指定されたクラス名をコンパイルします。

- Class—コンパイルするクラスの名前。
- Flags—コンパイル・フラグ [ck]。
- IgnoreErrors—エラー時に続行するかどうかを指定します[0]。

```
<Compile
    Class="MyPackage.MyClass"
    Flags="ck"
    IgnoreErrors=0/>
```

<Configuration>

親タグ：〈Namespace〉

子タグ： 〈ClassMapping〉、〈Database〉、〈GlobalMapping〉、〈RoutineMapping〉

〈Namespace〉 内で構成タグの親タグとして記述する必要があります。終了タグ (</Configuration>) は、ネームスペース内のデータベースのマッピングを有効化して、.cpf ファイルを更新します。

プロパティはありません。

```
<Configuration>
    <Database> . . . />
    <ClassMapping> . . . />
/>
```

<CopyClass>

親タグ：〈Namespace〉

ソース・クラス定義をターゲットにコピーまたは移動します。

- Src—ソース・クラス。
- Target—ターゲット・クラス。
- Replace—ターゲット・クラスを上書きするかどうかを指定します[0]。

```
<CopyClass
    Src="MyPackage.MyClass"
    Target="NewPackage.NewClass"
    Replace="0"/>
```

<CopyDir>

親タグ：〈Manifest〉

ソース・ディレクトリをターゲットにコピーします。

- Src—ソース・ディレクトリ。
- Target—ターゲット・ディレクトリ。

- IgnoreErrors—エラー時に続行するかどうかを指定します[0]。

```
<CopyDir
    Src="${MGRDIR}"
    Target="F:\MyTargetDir"
    IgnoreErrors="0"/>
```

<CopyFile>

親タグ：〈Manifest〉

ソース・ファイルをターゲットにコピーします。

- Src—ソース・ファイル。
- Target—ターゲット・ファイル。
- IgnoreErrors—エラー時に続行するかどうかを指定します[0]。

```
<CopyFile
    Src="${MGRDIR}\messages.log"
    Target="F:\${INSTANCE}_log"
    IgnoreErrors="0"/>
```

<CSPApplication>

親タグ：〈Namespace〉

Security.Applications クラス内で定義されている 1 つ以上の Web アプリケーションを定義します(これらの各フィールドの詳細は、"アプリケーションの作成および編集" も参照してください)。

- AuthenticationMethods—有効な認証方法。(サポートされている認証方法と、対応する提供値については、"Security.Applications" の "AutheEnabled" プロパティを参照してください)。例えば、一般的に使用される値として、4=Kerberos、32=password、64=unauthenticated があります。
- AutoCompile—(CSP 設定で) 自動コンパイルするかどうかを指定します[1]。
- CSPZENEnabled—CSP/ZEN を有効にするかどうかを指定します[1]。
- ChangePasswordPage—パスワード変更ページへのパス。
- CookiePath—セッション cookie パス。
- CustomErrorPage—カスタム・エラー・ページへのパス。
- DefaultSuperclass—デフォルト・スーパークラス。
- DefaultTimeout—セッション・タイムアウト。
- Description—説明。
- Directory—CSP ファイルへのパス。
- EventClass—イベント・クラス名。
- Grant—システムにログインするときに割り当てられるロールのリスト。
- GroupById—ID でグループ化するかどうかを指定します。
- InboundWebServicesEnabled—着信 Web サービスを有効にするかどうかを指定します[1]。
- IsNamespaceDefault—ネームスペースの既定のアプリケーションかどうかを指定します[0]。
- LockCSPName—CSP名をロックするかどうかを指定します[1]。
- LoginClass—ログイン・ページへのパス。

- PackageName—パッケージ名。
- PermittedClasses—許可されるクラス。
- Recurse—繰り返す (サブディレクトリを提供する) かどうかを指定します[0]。
- Resource—Web アプリケーションにアクセスするために必要なリソース。
- ServeFiles—ファイルを提供するかどうかを指定します。[1]。
  - 0—提供しない
  - 1—常に提供する
  - 2—常に提供し、キャッシュする
  - 3—CSP セキュリティを使用する
- ServeFilesTimeout—静的ファイルをキャッシュする時間を秒単位で指定します。
- TwoFactorEnabled—2 ステップ認証を有効にするかどうかを指定します[0]。
- Url—Web アプリケーションの名前。
- UseSessionCookie—セッションに cookie を使用するかどうかを指定します。

```
<CSPApplication
    Url="/csp/foo/bar"
    Description=""
    Directory="C\InterSystems\IRIS\CSP\Democode1"
    Resource=""
    Grant="%DB_%DEFAULT"
    Recurse="1"
    LoginClass=""
    CookiePath="/csp/demo1"
    AuthenticationMethods="64"/>
```

<Credential>

親タグ：〈Production〉

アクセス資格情報を作成またはオーバーライドします。

- Name—アクセス資格情報の名前。
- Username—ユーザ名。
- Password—ユーザ・パスワード。
- Overwrite—アカウントが既に存在する場合は上書きします。

```
<Credential
    Name="Admin"
    Username="administrator"
    Password="123jUgT540!f3B$#"
    Overwrite="0"/>
```

<Database>

親タグ：〈Configuration〉

ネームスペース内でデータベースを定義します。

以下のプロパティで制御するデータベース設定の詳細は、"システム管理ガイド" の “InterSystems IRIS の構成” の章にある "データベースの構成" を参照してください。

- BlockSize—データベースのブロック・サイズ (4096、8192、16384、32768、65536)。

- ClusterMountMode—起動時にクラスタの一部としてデータベースをマウントするかどうかを指定します。
- Collation—データベース内に作成される、グローバルの既定の照合。
- Create—新しいデータベースを作成するかどうかを指定します (yes/no/overwrite)[yes]
- Dir—データベース・ディレクトリ。
- Encrypted—データベースを暗号化するかどうかを指定します。
- EncryptionKeyID—暗号化キーの ID。
- InitialSize—データベースの初期サイズを MB 単位で指定します。
- ExpansionSize—必要な場合にデータベースを拡張するサイズを MB 単位で指定します。
- MaximumSize—データベースを拡張できる最大サイズ。
- MountAtStartup—インストールされたインスタンスを起動するときにマウントするかどうかを指定します。
- MountRequired—インスタンスを起動するたびにデータベースをマウントする必要があるかどうかを指定します。
- Name—データベース名。
- PublicPermissions—リソースを作成する必要がある場合、リソースに割り当てられる許可の値。リソースが既に存在する場合は無視されます。読み取り専用または読み書き可能です。
- Resource—データベースへのアクセスを制御するリソース。
- StreamLocation—データベースに関連付けられているストリームが格納されるディレクトリ。

```
<Database Name="${DBNAME}"
    Dir="${MGRDIR}/${DBNAME}"
    Create="yes"
    Resource="${DBRESOURCE}"
    Blocksize="8192"
    ClusterMountMode="0"
    Collation="5"
    Encrypted="0"
    EncryptionKeyID=
    ExpansionSize="0"
    InitialSize="1"
    MaximumSize="0"
    MountAtStartup="0"
    MountRequired="0"
    StreamLocation=
    PublicPermissions=""/>
<Database Name="MYAPP"
    Dir="${MYAPPDIR}/db"
    Create="no"
    Resource="${MYAPPRESOURCE}"
    Blocksize="8192"
    ClusterMountMode="0"
    Collation="5"
    Encrypted="0"
    EncryptionKeyID=
    ExpansionSize="0"
    InitialSize="1"
    MaximumSize="0"
    MountAtStartup="0"
    MountRequired="0"
    StreamLocation=
    PublicPermissions=""/>
```

<Default>

親タグ：〈Manifest〉

変数値がまだ設定されていない場合に、変数値を設定します。

- Name—変数名。

- Value—変数値。
- Dir—フォルダまたはファイルのパスを示す変数値。

```
<Default Name="blksiz"
    Value="8192" />
```

<Else>

親タグ：〈If〉

先行する <If> 文で定義される条件文が false の場合に実行されます。

プロパティはありません。

```
<If Condition='#
{##class(%File).Exists(INSTALL_"mgr\iris.key")}
'>
<Else/>
<CopyFile Src="C:\\InterSystems\key_files\iris300.key" Target="${MGRDIR}\iris.key"
IgnoreErrors="0"/>
</If>
```

<Error>

親タグ：〈Manifest〉

子タグ：〈Arg〉

エラーを生成します。

- Status：エラー・コード。
- Source：エラーの発生源。

```
<Error Status="$$$NamespaceDoesNotExist" Source=>
    <Arg Value="${NAMESPACE}"/>
</Error>
```

<ForEach>

親タグ：〈Manifest〉

指定されたインデックス・キーの繰り返しに使用する値を定義します。

- Index—変数名。
- Values—変数値のリスト。

```
<ForEach
    Index="TargetNameSpace"
    Values="%SYS,User">
    <!--Code for each iteration of TargetNameSpace-->
</ForEach>
```

<GlobalMapping>

親タグ：〈Configuration〉

グローバルを現在のネームスペースにマップします。

- Global：グローバル名。
- From：グローバルのソース・データベース。

- Collation：グローバル照合 [IRIS Standard]。

```
<GlobalMapping Global="MyAppData.*"
    From="MYAPP" Collation="30"/>
<GlobalMapping Global="cspRule"
    From="MYAPP"/>
```

<If>

親タグ：〈Manifest〉、〈Namespace〉

子タグ：〈Else〉

条件付きアクションを指定します。

- Condition：条件文。

```
<If Condition='$L("${NAMESPACE}")=0'>
    <Error Status="$$$NamespaceDoesNotExist" Source=>
        <Arg Value="${NAMESPACE}"/>
    </Error>
</If>
```

<IfDef>

親タグ：〈Manifest〉、〈Namespace〉

変数が設定されている場合の条件付きアクションを指定します。

- Var：変数名。

```
<IfDef Var="DBCreateName">
   <Database Name="${DBNAME}"
       Dir="${MGRDIR}/${DBNAME}"
       Create="yes"
       ...
</IfDef>
```

<IfNotDef>

親タグ：〈Manifest〉、〈Namespace〉

変数が設定されていない場合の条件付きアクションを指定します。

- Var：変数名。

<Import>

親タグ：〈Namespace〉

%SYSTEM.OBJ.ImportDir(File,Flags,Recurse) または %SYSTEM.OBJ.Load(File,Flags) を呼び出してファイルをインポートします。

- File—インポートするファイルまたはフォルダ。
- Flags—コンパイル・フラグ [ck]。
- IgnoreErrors—エラー時に続行するかどうかを指定します[0]。
- Recurse—再帰的にインポートするかどうかを指定します[0]。

<Invoke>

親タグ：〈Namespace〉

子タグ：〈Arg〉

クラス・メソッドを呼び出して、実行結果を変数の値として返します。

- Class—クラス名。
- Method—メソッド名。
- CheckStatus—返されたステータスをチェックするかどうかを指定します。
- Return—結果の書き込み先の変数の名前。

```
<Invoke Class="SECURITY.SSLConfigs" Method="GetCertificate" CheckStatus="1" Return="CertContents">
    <Arg Value="iris.cer"/>
</Invoke>
```

<LoadPage>

親タグ：〈Namespace〉

%SYSTEM.CSP.LoadPage(PageURL,Flags) または %SYSTEM.CSP.LoadPageDir(DirURL,Flags) を呼び出して CSP ページをロードします。

- Name—CSP ページの URL。
- Dir—CSP ページを含むディレクトリの URL。
- Flags—コンパイル・フラグ [ck]。
- IgnoreErrors—エラー時に続行するかどうかを指定します[0]。

<Log>

親タグ：〈Manifest〉

setup() メソッドで指定されたログのレベルが level プロパティの値以上の場合に、setup() メソッドで指定されたログにメッセージを記録します。

- Level—ログのレベル。-1 (none) から 3 (verbose) までの範囲です。
- Text—ログ・メッセージ (最大 32,000 文字の文字列)。

詳細は、"マニフェスト・ログへの書き込み" を参照してください。

<Manifest>

親タグ：なし (ルート・タグ。他のすべてのタグを含みます)

子タグ：〈CopyDir〉、〈CopyFile〉、〈Default〉、〈Else〉、〈Error〉、〈ForEach〉、〈If〉、〈IfDef〉、〈IfNotDef〉、〈Log〉、〈Namespace〉、〈Resource〉、〈Role〉、〈SystemSetting〉、〈User〉、〈Var〉

プロパティはありません。

```
<Manifest>
    <Namespace ... >
        <Configuration>
            <Database .../>
            <Database .../>
        </Configuration>
    </Namespace>
</Manifest>
```

<Namespace>

親タグ：〈Manifest〉

子タグ：〈Compile〉、〈Configuration〉、〈CopyClass〉、〈CSPApplication〉、〈Else〉、〈If〉、〈IfDef〉、〈IfNotDef〉、〈Import〉、〈Invoke〉、〈LoadPage〉、〈Production〉

ネームスペースを定義します。

- Name—ネームスペースの名前。
- Create—新しいネームスペースを作成するかどうかを指定します (yes/no/overwrite)[yes]
- Code—コード用データベース。
- Data—データ用データベース。
- Ensemble—相互運用対応のネームスペースかどうかを指定します[0]。

(相互運用 Web アプリケーションには他のプロパティを適用可能)

```
<Namespace Name="${NAMESPACE}"
     Create="yes"
     Code="${NAMESPACE}"
     Data="${NAMESPACE}">
          <Configuration>
               <Database Name="${NAMESPACE}" . . . />
          </Configuration>
</Namespace>
```

<Production>

親タグ：〈Namespace〉

子タグ：〈Credential〉、〈Setting〉

プロダクションを定義します。

- Name—プロダクション名。
- AutoStart—プロダクションを自動的に起動するかどうかを指定します[0]。

```
<Production Name="${NAMESPACE}"
     AutoStart="1" />
```

<Resource>

親タグ：〈Manifest〉

リソースを定義します。

- Name—リソース名。
- Description—リソースの説明。
- Permission—パブリック許可。

```
<Resource
    Name="%accounting_user"
    Description="Accounting"
    Permission="RW"/>
```

<Role>

親タグ：〈Manifest〉

ロールを定義します。

- Name—ロール名。
- Description—ロールの説明 (コンマは使用できません)。

- Resources—ロールで保持されている特権 (リソースと特権のペア)。
- RolesGranted—指定されたロールで付与されるロール。

```
<Role
    Name="%DB_USER"
    Description="Database user"
    Resources="MyResource:RW,MyResource1:RWU"
    RolesGranted= />
```

<RoutineMapping>

親タグ：〈Configuration〉

ルーチン・マッピングを定義します。

- Routines：ルーチン名。
- Type：ルーチン・タイプ (MAC、INT、INC、OBJ、ALL)。
- From：ルーチンのソース・データベース。

```
<RoutineMapping Routines="MyRoutine"
    Type="ALL" From="${NAMESPACE}" />
```

<Setting>

親タグ：〈Production〉

Ens.Production.ApplySettings() メソッドを呼び出して、プロダクション内のアイテムを構成します。

- Item—アイテム名。
- Target—設定タイプ (Item、Host、Adapter)。
- Setting—設定名。
- Value—設定の構成値。

```
<Production Name="Demo.ComplexMap.SemesterProduction">
    <Setting Item="Semester_Data_FileService"
        Target="Item"
        Setting="PoolSize"
        Value="1"/>
    <Setting Item="Semester_Data_FileService"
        Target="Host"
        Setting="ComplexMap"
        Value="Demo.ComplexMap.Semester.SemesterData"/>
    <Setting Item="Semester_Data_FileService"
        Target="Adapter"
        Setting="FilePath"
        Value="C:\Practice\in\"/>
</Production>
```

<SystemSetting>

親タグ：〈Manifest〉

Config.CommonSingleMethods から Modify() メソッドを継承する任意の Config クラスのプロパティの値を設定します。

- Name—Config クラスの Class.property。
- Value—プロパティに割り当てる値。

<User>

親タグ：〈Manifest〉

ユーザを定義します。PasswordVar が含まれる場合、指定された変数名でユーザのパスワードを渡す必要があります。

- Username—ユーザ名。
- PasswordVar—ユーザ・パスワードが含まれる変数の名前 (以下の 〈Var〉 を参照)。
- Roles—ユーザを割り当てるロールのリスト。
- Fullname—ユーザのフルネーム。
- Namespace—ユーザの実行開始ネームスペース。
- Routine—ユーザの実行開始ルーチン。
- ExpirationDate—それ以降のユーザ・ログインが無効になる日付。
- ChangePassword—次回ログイン時にユーザにパスワードの変更を要求するかどうかを指定します。
- Enabled—ユーザが有効かどうかを指定します。
- Comment—オプションコメント。

```
<User
    Username="Clerk1"
    PasswordVar="clerk1pw"
    Roles="Dataentry"
    Fullname="Data Entry Clerk"
    Namespace=
    Routine=
    ExpirationDate=
    ChangePassword=
    Enabled=
    Comment=""/>
```

<Var>

親タグ：〈Manifest〉

マニフェストで使用できる変数を定義および設定します。

- Name—変数名。
- Value—変数に割り当てる値。

```
<Var Name="Namespace" Value="MUSIC"/>
<Var Name="GlobalDatabase" Value="${Namespace}G"/>
<Var Name="RoutineDatabase" Value="${Namespace}R"/>
<Var Name="AppDir" Value="C:\MyApp\${CFGNAME}"/>
<Var Name="GlobalDatabaseDir" Value="${AppDir}\${GlobalDatabase}"/>
<Var Name="RoutineDatabaseDir" Value="${AppDir}\${RoutineDatabase}"/>
<Var Name="Resource" Value="%DB_${Namespace}"/>
<Var Name="Role" Value="${Namespace}"/>
<Var Name="CSPResource" Value="CSP_${Namespace}"/>
```

5.3 マニフェストの使用

マニフェストは、手動で実行することも、インストールの一部として実行することもできます。

5.3.1 手動で実行する場合

%SYS ネームスペースで、ターミナルに以下のコマンドを入力します。

```
%SYS>do ##class(MyPackage.MyInstaller).setup()
```

変数の配列を参照として setup() メソッドに渡すことができます。各添え字は変数名として使用され、ノードは値として使用されます。以下に例を示します。

```
%SYS>set vars("SourceDir")="c:\myinstaller"
%SYS>set vars("Updated")="Yes"
%SYS>do ##class(MyPackage.MyInstaller).setup(.vars,3)
```

この例の 2 番目の引数 (3) は、ログ・レベルです。

5.3.2 インストール時に実行する場合

マニフェスト・クラスを DefaultInstallerClass.xml として InterSystems IRIS インストール (.msi、setup_irisdb.exe、または irisinstall) が実行されている場所と同じディレクトリにエクスポートします。これは %SYS にインポートされてコンパイルされ、setup() メソッドが実行されます。

注釈　インストールを実行するユーザが、マニフェスト・ログ・ファイル・ディレクトリへの書き込み許可を持っていることを確認します。ログ・ファイルに書き込めない場合、インストールは続行されません。

また、単一ファイル・キットを使用してインストールしている場合、想定どおりに動作するようにするには、ファイルを .exe ファイルからインストール・ディレクトリに抽出する必要があります。

エクスポートする方法では、引数を setup() メソッドに直接渡せないことに注意してください。以下のセクションでは、Windows で、または UNIX®、Linux、および macOS で引数を渡す方法について説明します。

5.3.2.1 Windows

.msi インストール・パッケージを変更して、変数の名前と値のペアを setup() メソッドに渡すことができます。

また、以下の例に示すようにインストーラでコマンド行引数を使用して、エクスポートされるマニフェスト・クラスの場所、変数、ログ・ファイル名、およびログのレベルを渡すこともできます。

```
setup.exe INSTALLERMANIFEST="c:\MyStuff\MyInstaller.xml"
INSTALLERMANIFESTPARAMS="SourceDir=c:\mysourcedir,Updated=Yes"
INSTALLERMANIFESTLOGFILE="installer_log" INSTALLERMANIFESTLOGLEVEL="2"
```

注釈　INSTALLERMANIFESTPARAMS 引数を使用して渡される変数名には、アルファベット文字と数字 (A-Za-z0-9)、およびアンダースコア (_) のみを含めることができますが、先頭をアンダースコアにすることはできません。

詳細は、"コマンド行のリファレンス" を参照してください。

5.3.2.2 UNIX®、Linux、および macOS

UNIX®、Linux、および macOS システムでは、以下の例に示すように、irisinstall または irisinstall_silent のいずれかを実行する前に環境変数を設定して、エクスポートされるマニフェスト・クラスの場所、変数、ログ・ファイル名、およびログのレベルを定義できます。

```
ISC_INSTALLER_MANIFEST="/MyStuff/MyInstaller.xml"
ISC_INSTALLER_PARAMETERS="SourceDir=/mysourcedir,Updated=Yes"
ISC_INSTALLER_LOGFILE="installer_log"
ISC_INSTALLER_LOGLEVEL="2"
./irisinstall
```

詳細は、"自動インストール・パラメータ" を参照してください。

5.4 インストール・マニフェストの例

このセクションでは、以下を行うインストール・マニフェストの例を示します。

- 空のテンプレート
- ネームスペースの作成
- アップグレード後のコンパイル

5.4.1 空のテンプレート

以下に、基本的なマニフェスト・テンプレートを示します。

Class Definition

```
Include %occInclude

/// Simple example of installation instructions (Manifest)
Class MyInstallerPackage.SimpleManifest
{

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
   <Log Level="3" Text="Start manifest" />
   <Namespace Name="">
      <Import File="" />
      <Invoke Class="" Method="" CheckStatus="" Return="">
         <Arg Value=""/>
      </Invoke>
   </Namespace>
   <CopyFile Src="" Target="" IgnoreErrors=""/>
   <CopyDir Src="" Target="" IgnoreErrors=""/>
   <Log Level="3" Text="End manifest" />
</Manifest>
}

/// This is a method generator whose code is generated by XGL.
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
{
    #; Let our XGL document generate code for this method.
    Quit ##class(%Installer.Manifest).%Generate(%compiledclass, %code, "SimpleManifest")
}

}
```

5.4.2 ネームスペースの作成

この例のマニフェストでは、ネームスペース (MyNamespace) と 3 つのデータベースを作成します。MyDataDB データベースと MyRoutinesDB データベースはそれぞれグローバルとルーチンの既定のデータベースで、MyMappingDB データベースは別のグローバル・データベースです。<GlobalMapping> タグは、MyNamespace ネームスペースにある t* グローバルに対して MyMappingDB へのマッピングを作成します。これは、既定のマッピングをオーバーライドします。

Class Definition

```
Include %occInclude

/// Simple example of installation instructions (Manifest)
Class MyInstallerPackage.SimpleManifest
```

```
{

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
    <Namespace Name="MyNamespace" Create="yes"
        Code="MyRoutinesDB" Data="MyDataDB">
      <Configuration>
        <Database Name="MyRoutinesDB" Create="yes"
            Dir="C:\MyInstallerDir\MyRoutinesDB"/>
        <Database Name="MyDataDB" Create="yes"
            Dir="C:\MyInstallerDir\MyDataDB"/>
        <Database Name="MyMappingDB" Create="yes"
            Dir="C:\MyInstallerDir\MyMappingDB"/>
        <GlobalMapping Global="t*" From="MyMappingDB"/>
      </Configuration>
    </Namespace>
</Manifest>
}

/// This is a method generator whose code is generated by XGL.
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
{
    #; Let our XGL document generate code for this method.
    Quit ##class(%Installer.Manifest).%Generate(%compiledclass,
    %code, "SimpleManifest")
}

}
```

5.4.3 アップグレード後のコンパイル

この例のマニフェストでは APPTEST という名前のネームスペース内のコードをリコンパイルし、アップグレード後に実行します。以下のタスクを実行します。

1. マニフェスト・インストール・プロセスの開始を示すメッセージをマニフェスト・ログに書き込みます。
2. 現在のネームスペースを APPTEST に設定します。これには、アップグレード後にリコンパイルする必要のあるコードが含まれます。
3. メソッド %SYSTEM.OBJ.CompileAll() を呼び出して、ネームスペース内のクラスをコンパイルします。
4. メソッド %Routine.CompileAll() を呼び出して、ネームスペース内のカスタム・ルーチンをコンパイルします。
5. マニフェスト・インストール・プロセスの終了を示すメッセージをマニフェスト・ログに書き込みます。

Class Definition

```
Include %occInclude

/// Simple example of installation instructions (Manifest)
Class MyInstallerPackage.SimpleManifest
{

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
    <Log Level="3" Text="The Installer Manifest is running."/>
    <Namespace Name="APPTEST">
        <Invoke Class="%SYSTEM.OBJ" Method="CompileAll" CheckStatus="1">
            <Arg Value="u"/>
        </Invoke>
        <Invoke Class="%Routine" Method="CompileAll" CheckStatus="1"/>
        </Namespace>
    <Log Level="3" Text="The Installer Manifest has completed."/>
</Manifest>
}

/// This is a method generator whose code is generated by XGL.
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
```

```
{
   #; Let our XGL document generate code for this method.
   Quit ##class(%Installer.Manifest).%Generate(%compiledclass, %code, "SimpleManifest")
}

}
```

6

複数の InterSystems IRIS インスタンスの構成

1 台のマシンに、複数のインスタンスをインストールして、同時に実行できます。それには、InterSystems IRIS の各インスタンスに一意の名前を付け、別々のディレクトリにインストールし、それぞれに異なるポート番号を割り当てます。

詳細は、"システム管理ガイド" の "複数の InterSystems IRIS インスタンス" のセクションを参照してください。

複数のインスタンスの実行時には、アプリケーションへのアクセスを構成するうえで追加の考慮事項があります。この詳細は、"複数の InterSystems IRIS サーバ上のターゲット・アプリケーション" を参照してください。

注釈　一度にインストールする必要がある InterSystems IRIS インスタンスは 1 つのみであるため、URL にインスタンス名を指定する必要がないユーザの利便性のため、インストーラは現在インストールしているインスタンスにルーティングするルートレベルのアプリケーション・パスを持つ URL を自動的に構成します。

ただしこのことは、複数のインスタンスが構成されたシステムで、最近構成されたインスタンスをアンインストールすると、ルートレベルのアプリケーション・パスに対する要求が成功しなくなることを意味しています。Web ゲートウェイのアプリケーション・アクセス・プロファイルで / (ルート) および /csp パスを手動で更新して、アンインストールしたインスタンス以外の既定のアプリケーション・サーバに要求を送信するようにする必要があります。その後、そのインスタンスの古いサーバ・アクセス・プロファイルを削除できます。

7

InterSystems IRIS の言語の変更

InterSystems IRIS をインストールすると、サポート対象言語固有のユーティリティ DLL は、すべて install-dir¥bin ディレクトリにインストールされます。各 DLL には、ローカライズされた文字列とメッセージが含まれます。

DLL 名の形式は **UTILaaa.DLL** で、aaa には、以下の言語を表す 3 文字のコードが入ります。

コード	言語
CHS	中国語 (簡体字)
DEU	ドイツ語 (標準)
ENU	英語 (アメリカ)
ESP	スペイン語 (スペイン)
FRA	フランス語
ITA	イタリア語 (標準)
JPN	日本語
KOR	韓国語
NLD	オランダ語 (標準)
PTB	ポルトガル語 (ブラジル)
RUS	ロシア語

InterSystems IRIS インストールのロケール変更の詳細は、"InterSystems IRIS システム管理ガイド" の “InterSystems IRIS の構成” の章の "管理ポータルの NLS 設定ページの使用法" を参照してください。

注釈　変更は 8 ビット・ロケール間または Unicode ロケール間でのみ可能で、8 ビットから Unicode、または Unicode から 8 ビットに変更することはできません。詳細は、"インターシステムズ・クラスリファレンス" の %SYS.NLS のエントリを参照してください。

8

Web サーバのセットアップのトラブルシューティング

InterSystems IRIS をインストールする前に Web サーバをインストールしている場合は、InterSystems IRIS インストール・プロセスで自動的に Web サーバが構成されます。ここでは、Web サーバのセットアップに対応するためのいくつかのトラブルシューティング手順を説明します。

8.1 ドキュメント・リンクが機能しない

管理ポータル (または Windows ランチャー) 内のドキュメントへのリンクが機能しない場合、ドキュメント・リンクをリダイレクトすることが必要な場合があります。Apache Web サーバと Microsoft IIS Web サーバでは、手順が異なります。

8.2 Windows と IIS

8.2.1 VS Code のトラブルシューティング

VS Code が InterSystems IRIS インスタンスに接続できるようにするには、追加の手順が必要になる場合があります。詳細は、"Windows のみ：VS Code を有効にするように IIS を構成" を参照してください。

VS Code は機能するが、デバッグは機能しない場合は、WebSocket 機能を有効化することが必要な場合があります。

8.2.2 IIS の再起動

一部の変更を有効にするには、IIS Web サーバを再起動する必要があります。インストール・プロセスの完了後を含め、Web サーバまたは Web ゲートウェイの構成に対する変更を行った後は、Web サーバを再起動することをお勧めします。IIS Web サーバを再起動するには、以下の手順を実行します。

1. Windows コマンド行インタフェースを使用し、以下のコマンドを実行して Web サーバを停止します。

   ```
   sc stop W3SVC
   ```

2. Web サーバが停止したら、以下のコマンドを実行して Web サーバを再起動します。

   ```
   sc start W3SVC
   ```

8.2.3 インスタンスの修復または変更

インストール・プロセス中に IIS Web サーバの自動構成がスキップされた場合は、インストール後にこれを行うことができます。そのためには、最初にインスタンスをインストールするために使用したのと同じ実行可能ファイルを使用して、インストールを実行します。[IIS用 CSP] コンポーネントが初期インストールに含まれていたかどうかに応じて、[**修復**] または [**変更**] を選択します。

[IIS用 CSP] コンポーネントが元々インストールされていたが、IIS が構成されていなかったか、構成が手動で削除された場合は、[**修復**] を実行して IIS Web サーバの自動構成を開始できます。

重要　　インスタンスを自動構成せずに [IIS用 CSP] コンポーネントを含める唯一の方法は、`CSPSKIPIISCONFIG=1` を指定することです。このフラグは、自動インストール、またはコマンド行からインストール・ファイルを実行する場合にのみ指定できます。インストール中にこのフラグを含めなかった場合は、[**修復**] を実行しないでください。代わりに、インスタンスの [**変更**] を実行して、自動構成を開始できます。

[IIS用 CSP] が元のインストールに含まれていなかった場合は、インストールの [**変更**] を実行して、[IIS用 CSP] コンポーネントを選択できます。これによりインストーラで IIS Web サーバの自動構成が開始されます。

どちらの場合も、インストールの終了後、すべての機能が適切に動作するために追加の構成手順を検証する必要があります。

1. 管理ポータルへのアクセスを有効にするには、サーバ・アクセス・プロファイル内でインスタンスの CSPSystem アカウント認証情報を設定します。詳細は、"自動構成動作" の手順 6 を参照してください。
2. ランチャーのリンクを有効にするには、以下の InterSystems IRIS サーバ・マネージャのパラメータを構成する必要があります。
   - [**ウェブサーバポート**] を、Web サーバで使用しているポート番号 (既定では 80) に設定します。
   - [**CSPサーバインスタンス**] を、小文字のインスタンス名 (`iris` など) に設定します。