InterSystems ODBC ドライバの使用法

Version 2024.1
2024-06-03

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

InterSystems ODBC ドライバの使用法
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

図一覧

このドキュメントについて

このドキュメントでは、InterSystems ODBC ドライバの使用方法について説明します。これを使用すると、外部アプリケーションから ODBC を介して InterSystems データベースに接続でき、インターシステムズ製品から外部 ODBC データ・ソースにもアクセスできます。

このドキュメントでは、以下の項目について説明します。

- はじめに：インターシステムズ・データベースへの ODBC 接続 ー InterSystems ODBC ドライバの概要を説明します。
- Windows での ODBC データ・ソースの定義 ー Windows で InterSystems IRIS を ODBC データ・ソースとして使用する方法について説明します。
- UNIX® での ODBC データ・ソースの定義 ー UNIX® で InterSystems IRIS を ODBC データ・ソースとして使用する方法について説明します。
- UNIX® システムでの ODBC のインストールと検証 ー ODBC インストールをテストおよび検証するツールについて説明し、スタンドアロン・インストールおよび iODBC のカスタム・インストールの手順についても説明します。
- Python および Node.js 用の ODBC サポート ー ODBC の Python および Node.js オープン・ソース・インタフェースのサポートについて説明します。
- ログと環境変数 ー トラブルシューティングを実行するために使用できるツールについて説明します。

詳細は、以下のソースを参照してください。

- ″InterSystems Managed Provider for .NET の使用法″ ー リレーショナル・データ・アクセスに ADO.NET Managed Provider、オブジェクト・アクセスに Entity Framework のインターシステムズの実装を使用する方法について説明します。
- ″InterSystems ソフトウェアでの Java の使用法″ には、外部データ・ソースから InterSystems IRIS への JDBC 接続に関する情報が記載されています (このドキュメントで説明している JDBC と同等の内容です)。
- ″InterSystems SQL ゲートウェイの使用法″ では、SQL ゲートウェイが ODBC と JDBC の両方でどのように動作するかについて概要が説明されています。

1

はじめに : インターシステムズ・データベースへの ODBC 接続

インターシステムズは、ODBC 接続を介した InterSystems データベースへのアクセスを可能にする ODBC ドライバを提供しています。ODBC を使用するには、InterSystems ODBC クライアント・ドライバをインストールおよび構成し、InterSystems データベースを参照するように 1 つ以上の DSN を定義します。他の DSN を使用する場合と同様の方法で、InterSystems DSN をアプリケーションで使用できるようになります。

″Connecting Your Application to InterSystems IRIS″ には、サンプル・コードを含め、ODBC を使用して C++ アプリケーションから InterSystems IRIS サーバに接続する手順が示されています。

1.1 インストール

InterSystems データベースを ODBC データ・ソースとして使用するには、最初に、InterSystems ODBC クライアント・ドライバがインストールされていることを確認する必要があります。以下のオプションを使用できます。

- インターシステムズの標準インストールでは、既定で ODBC ドライバ・コンポーネントがインストールされます (″インストール・ガイド″ を参照してください)。
- カスタム・インストールを実行する場合は、[SQL] オプションを選択して、ODBC クライアント・ドライバのみをインストールできます。
- InterSystems IRIS がシステムにインストールされていない場合は、Windows、Linux、または macOS 用のドライバを InterSystems IRIS ドライバのページからダウンロードできます。
- その他のオプションは、一部の UNIX® ドライバ・マネージャで利用可能です (“UNIX® システムでの ODBC のインストールと検証” を参照)。

また、DSN (データ・ソース名) を定義して、InterSystems データベースへの接続に必要な情報を ODBC 対応アプリケーションに提供する必要もあります。各 InterSystems データベースは複数の DSN で表現でき、それぞれが複数の接続をサポートできます。これらのタスクを実行するための OS 固有の手順については、“Windows での ODBC データ・ソースの定義” または “UNIX®での ODBC データ・ソースの定義” を参照してください。

注釈　Windows では、InterSystems IRIS ID に大型の数値 (**BigInt**) データ型を使用するため、ODBC クライアント・アプリケーションは大型の数値をサポートしている必要があります。例えば、ビルド 16.0.7812 より前の Access 2016 のインスタンスでは、行データは `#Deleted` と表示されます。これは、現在のデータベースの Access の設定で大型の数値のサポートがオンになっていない場合にも発生する場合があります。

1.2 サポートされている ODBC ドライバ・マネージャ

InterSystems ODBC ドライバは、ODBC 3.5 に準拠しています。

InterSystems ODBC は、以下の ODBC ドライバ・マネージャをサポートしています。

- Windows の場合 : オペレーティング・システム付属の Microsoft Windows ドライバ・マネージャ
- UNIX® の場合 : iODBC ドライバ・マネージャ (Unicode の ODBC API と 8 ビットの ODBC API で使用) および unixODBC ドライバ・マネージャ (8 ビットの ODBC API で使用)詳細は、“UNIX® システムでの ODBC のインストールと検証”を参照してください。

  他のドライバ・マネージャに関する質問については、インターシステムズのサポート窓口 (WRC) までお問い合わせください。

インターシステムズでは、これらのドライバ・マネージャと連携する Python および Node.js のオープン・ソース・モジュールもサポートしており、言語固有の ODBC インタフェースを提供しています。“Python および Node.js 用の ODBC サポート”の以下のセクションを参照してください。

- pyodbc Python ODBC ブリッジのサポート
- Node.js リレーショナル・アクセスのサポート

サポートされているデータベースなどの詳細は、"インターシステムズのサポート対象プラットフォーム" を参照してください。

1.3 ODBC の概要

ODBC システムは、以下のコンポーネントで構成されています。

- クライアント・アプリケーション – Microsoft の ODBC API に従って呼び出しを実行するアプリケーション。ODBC 呼び出しによってクライアントからデータ・ソースへの接続が確立します (“ODBC 接続の詳細”のセクションを参照)。
- ODBC ドライバ・マネージャ – ODBC API を使用してアプリケーションからの呼び出しを受け取り、これを登録済みの ODBC クライアント・ドライバに渡します。また、ドライバ・マネージャは、クライアント・アプリケーションがクライアント・ドライバと、さらにはデータベース・サーバと通信するために、必要なタスクを実行します。
- ODBC クライアント・ドライバ – データベース固有のアプリケーションで、ODBC ドライバ・マネージャを通してクライアント・アプリケーションからの呼び出しを受け取り、データベース・サーバとの通信を提供します。また、アプリケーションから要求された ODBC 関連データ変換を実行します。
- データベース・サーバ – クライアント・アプリケーションからの呼び出しを最終的に受け取る、実際のデータベース。これは、呼び出し元であるクライアント・ドライバと同じマシンまたは異なるマシンのいずれに置いてもかまいません。
- 初期化ファイル – ドライバ・マネージャの一連の構成情報。オペレーティング・システムによっては、クライアント・ドライバ情報を含むものもあります。UNIX® では **odbc.ini** という実際のファイルです。Windows ではレジストリ・エントリです。

注釈　特定のベンダのデータベースでは、ベンダからそのプラットフォーム専用のバージョンの ODBC クライアント・ドライバが提供されている場合があります。例えば、Oracle からは、Oracle データベースを Windows で使用するための専用の ODBC ドライバが提供されています。場合によっては、このようなドライバを使用するのが最適です。ベンダのドライバは、データベースの機能を最大限に活用してパフォーマンスの最適化または信頼性の向上を実現しているためです。

1.3.1 ODBC 接続の詳細

ODBC を介してデータベースに接続する場合は、通常、以下の接続情報を提供する必要があります。

- 使用する ODBC クライアント・ドライバに関する情報。
- データベースの検索およびアクセスに関する情報。例えば、データベースが存在するサーバや、データベース接続に使用されるポートなどの情報が含まれます。必要な情報は、該当するデータベース・テクノロジに応じて異なります。
- データベースがパスワードによって保護されている場合は、データベースにアクセスするためのログイン資格情報。

ほとんどの場合、この情報は DSN に格納され、これはクライアント・アプリケーションで使用するための論理名を持ちます。DSN には、ログイン資格情報が含まれている場合も、含まれていない場合もあります。またログイン資格情報は、データベース初期化ファイルに格納することも、一切格納しないこともできます。

DSN は ODBC ドライバ・マネージャに登録する必要があります。

接続は、以下のように確立されます。

1. クライアント・アプリケーションには、ODBC 呼び出しが含まれ、特定の DSN へ接続しようと試みます。クライアント・アプリケーションにリンクされている ODBC ドライバ・マネージャが、その呼び出しを受け取ります。
2. ODBC ドライバ・マネージャは、初期化ファイルを読み取って ODBC クライアント・ドライバの位置を取得し、クライアント・ドライバをメモリにロードします。
3. ODBC クライアント・ドライバがメモリにロードされると、ODBC 初期化ファイルを使用して DSN への接続情報やその他の情報の位置を確認します。この情報を使用して、クライアント・ドライバは指定されたデータベースに接続します。
4. 接続を確立した後、クライアント・ドライバはデータベース・サーバとの通信を管理します。

注意　DSN パスワードが指定される場合、これらはプレーン・テキストで保存されます。そのため、DSN の作成時にパスワードがホスト・マシンに保存される場合、このパスワードが他のユーザに見える可能性があります。インターシステムズは、パスワードの保存はお勧めしていません。これは、リスク評価が行われた場合にのみ行うべきです。

2

Windows での ODBC データ・ソースの定義

この章では、コントロール・パネルを使用するか DSN ファイルを作成することにより、Windows で InterSystems データベース用の DSN を作成する方法について説明します。

インターシステムズ・データベース用に DSN を作成するには、Windows 向け Intersystems ODBC ドライバをインストールする必要があります。操作しているホストに InterSystems IRIS がインストールされている場合、このドライバは既にインストールされています。そうでない場合は、InterSystems IRIS ドライバのページから InterSystems IRIS ODBC ドライバをダウンロードし、ダウンロードしたファイルを実行します。

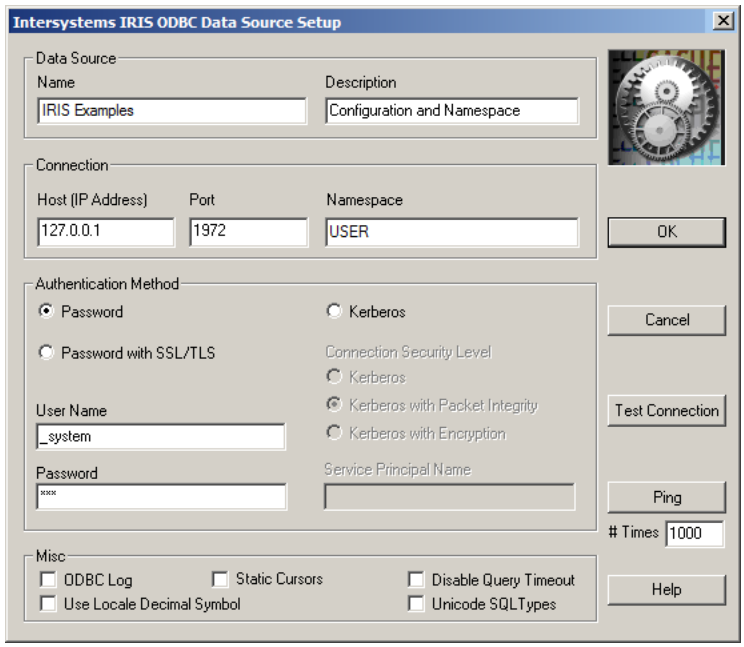
2.1 ODBC データ ソース アドミニストレーターによる DSN の作成

DSN を作成するには、Windows の [ODBC データ ソース アドミニストレーター] を使用して [InterSystems ODBC データソース設定] ダイアログ・ボックスにアクセスできます。

- Windows の [コントロール パネル] で、[ ] を選択して ODBC のアイコンをクリックします (実際のアイコン名は Windows のバージョンによって異なる場合があります。以下の “ODBC データ ソース アドミニストレーターの正しいバージョンの選択” を参照してください)。
- Windows の [ODBC データ ソース アドミニストレーター] ダイアログで、[DSN] タブを選択し、[...] ボタンをクリックします。
- ODBC 2.5 ドライバの場合は [Intersystems IRIS ODBC]、ODBC 3.5 ドライバの場合は [Intersystems IRIS ODBC35] を選択し、[ ] ボタンをクリックします。

以下の図は、すべての必須フィールドに入力した状態の [InterSystems ODBC データソース設定] ダイアログ・ボックスのインスタンスを示しています。

図 2-1: [InterSystems ODBC データソースセットアップ] ダイアログ・ボックス



各フィールドは以下のとおりです。特に指定がない限り、それらのフィールドは必須項目です。

[データソース] および [接続] のセクション

- [ ] – DSN のユーザ定義名を指定します。
- [ ] – オプション。DSN に関するユーザ定義の情報を入力します。
- [(IP)] – ODBC 接続で使用する IP アドレスを指定します。“127.0.0.1” のように、ドットで区切られた 10 進法の数字、あるいは 4 個の数字で表します。
- [ ] – ODBC 接続で使用するポートを指定します (″構成パラメータ・ファイル・リファレンス″ の “DefaultPort” を参照)。
- [ ] – ODBC データ・ソースとして使用するネームスペースを指定します。

[認証方法] セクション

- [ ] – このデータベースに使用されているセキュリティに応じて、以下のオプションのいずれかを選択します。これらのオプションの詳細は、″認証ガイド″ を参照してください。
  - [ ] – 標準のユーザ名とパスワードで認証します。
  - [SSL/TLS] – SSL/TLS で保護された接続を使用して認証します (″TLS ガイド″ を参照)。
  - [Kerberos] – Kerberos を使用して認証します (″Kerberos 認証″ を参照)。このオプションでは、以下の設定も指定します。
    - [ ] – [Kerberos]、[Kerberos]、または [Kerberos] を適宜選択します (″Kerberos およびアクセス・モードについて″ の ″クライアント・サーバ″ のセクションを参照してください)。
    - [ ] – Kerberos プリンシパルとして使用するサーバの名前を指定します。

- [　　] ー オプション。ODBC 接続で使用するユーザ名を指定します。既定では、これは _SYSTEM です (大文字と小文字は区別されません)。
- [　　] ー オプション。ODBC 接続で使用するパスワードを指定します。既定のユーザ名の場合、パスワードは SYS です (すべて大文字にする必要があります)。

[その他] セクション (オプション設定)

- [ODBC　　] ー オプション。選択されている場合は、すべての InterSystems DSN について、ODBC クライアント・ドライバの動作のログ・ファイルを作成します。このログは、問題が発生したときに利用します。これをオンにすると ODBC の動作速度が著しく低下するため、通常の運用時にはチェックを付けません。詳細は、“Windows の ODBC ログ” を参照してください。
- [　　] ー オプション。チェックが付いている場合は、InterSystems ODBC クライアント・ドライバのスタティック・カーソル・サポートを有効にします。チェックが付いていない場合は、ODBC カーソル・ライブラリのカーソル・サポートを使用します。通常、ODBC カーソル・ライブラリを使用しない特別な理由がない限りは、チェックを付けません。
- [　　] ー オプション。チェックが付いている場合は、ODBC クライアント・ドライバが ODBC クエリ・タイムアウトの設定値を無視します。

  ODBC クエリ・タイムアウト設定は、特定の動作が完了するまでにクライアントが待機する時間を指定します。動作が指定時間内に完了しなかった場合、自動的にキャンセルされます。ODBC API には、このタイムアウト値をプログラムで設定する機能があります。ただし、一部の ODBC アプリケーションではこの値をハード・コーディングします。ご使用の ODBC アプリケーションでタイムアウト値を設定できず、そのタイムアウト値が小さすぎる場合、クエリ・タイムアウトを無効にするオプションにより、タイムアウト値を無効にできます。
- [　　] ー オプション。チェックを付けると、現在のロケールの小数点区切り文字を使用することを指定します。チェックを外すと、ロケールに関係なく、プロセスの小数点区切り文字はピリオド (".") に設定されます。この値は、現在のロケールに定義された小数点区切り文字を使用するアプリケーションと ODBC 接続が相互運用する際に影響を与える可能性があります。
- [Unicode SQL　　] ー オプション。中国語、ヘブライ語、日本語、韓国語などのロケールで、マルチバイトの文字セットを使用する場合にのみ、この機能が関係します。シングル・バイトの文字セットのデータのみを使用する場合は、このチェック・ボックスにチェックを付けないでください。チェックが付いている場合、このオプションは、文字列データの Unicode SQL タイプ (SQL_WVARCHAR (-9) SQLType) のレポート機能をオンにします。これにより、一部の Microsoft アプリケーションはマルチバイトのデータを保持するために、適切なサイズのバッファを割り当てます。

  このチェック・ボックスにチェックが付いている場合、SQLBindParameter を使用した際に、アプリケーションでは Microsoft ドライバ・マネージャの “SQL データ型が範囲を越えています” というエラーが発生する可能性があります。

DSN を作成した後、[　　] ボタンを使用して、データ・ソースが正常に動作しているかどうかを確認できます。

[Ping] ボタンをクリックすると、[　] フィールドで指定した回数、DSN ホスト・マシンへの Ping を試行します。Ping の成功または失敗に関する情報がポップアップ・ウィンドウに表示されます。

注釈　**Windows Power Shell コマンド**

Windows では、コマンド行から DSN を操作するための Power Shell コマンドのセットも提供されています。詳細は、Windows Data Access Components (WDAC) の Power Shell のドキュメントを参照してください。

2.1.1 ODBC データ ソース アドミニストレーターの正しいバージョンの選択

64 ビットの Windows でユーザ DSN を構成するには、32 ビットと 64 ビット両方のプログラムに対して、Windows の [コントロール パネル] の [ODBC アドミニストレーター] を使用します。

32 ビット・プログラムにシステム DSN を構成するには、%SystemRoot%¥SysWow64¥odbcad32.exe を実行します。

2.2 ファイル DSN 接続および DSN なし接続の使用法

DSN 情報は通常、Windows レジストリ ([HKLM¥SOFTWARE¥ODBC]) に保存されますが、ファイル DSN (拡張子 .dsn のテキスト・ファイル) で接続情報を指定することもできます。

ファイル DSN は、[ODBC データ ソース アドミニストレーター] ([DSN] タブ) または標準のテキスト・エディタで作成できます。詳細は、Microsoft サポート・サイトを参照してください (" DSN" で検索してください)。

ファイル DSN では、使用する既存の DSN の名前を指定できます。以下に例を示します。

```
[ODBC]
DSN=InterSystems ODBC Sample Code
```

または、キーと値のペア (標準のレジストリ・エントリと同じ接続情報を指定します) のセットを指定できます。

ファイル DSN は、SQLDriverConnect の呼び出しによって呼び出します。

ファイル DSN は通常、¥Program Files¥Common Files¥ODBC¥Data Sources に保存されますが、[ODBC データ ソース アドミニストレーター] の [DSN] タブを使用して、別の既定の場所を定義することもできます。

2.2.1 ODBC 接続文字列

SQLDriverConnect は接続文字列引数を取り、この引数によって、以下の 3 つの異なる方法で接続情報を指定できます。

DSN 接続

レジストリ内の標準の DSN の名前を指定します。以下に例を示します。

```
"DSN=ODBC Samples;UID=myUsername;PWD=;"
```

FILEDSN 接続

レジストリ・エントリの代わりにファイル DSN を指定します。以下に例を示します。

```
"FILEDSN=c:\ODBC_Samples.dsn;UID=myUsername;PWD=;"
```

DSN なし接続

すべての接続情報を直接、接続文字列で定義します。以下に例を示します。

```
"Driver=InterSystems ODBC Driver;Host=127.0.0.1;Port=51774;Database=USER;UID=myUsername;PWD="
```

3

UNIX® での ODBC データ・ソースの定義

外部アプリケーションから InterSystems データベースを ODBC データ・ソースとして使用することができます。この章では、UNIX® で InterSystems データベース用の DSN を作成する方法について説明します。これは、ODBC 初期化ファイルを編集することによって行います。

インターシステムズ・データベース用に DSN を作成するには、UNIX 向け Intersystems ODBC ドライバをインストールする必要があります。操作しているホストに InterSystems IRIS がインストールされている場合、このドライバは既にインストールされています。そうでない場合は、以下を実行します。

1. InterSystems IRIS ドライバのページから InterSystems ODBC ドライバをダウンロードします。
2. ドライバのインストール先のディレクトリ（例：/usr/irisodbc）を作成し、ダウンロードした .tar ファイルをそのディレクトリでアンパックします。
3. ODBCinstall スクリプトを実行します。

3.1 ODBC 初期化ファイルの構造

ODBC 初期化ファイルの役割は以下のとおりです。

- 特定の接続に必要な ODBC クライアント・ドライバへのパスを含めて、情報を提供します。これによって、利用可能な DSN をドライバ・マネージャが確認して接続できます。
- DSN を定義します（オプションで、それらのログイン資格情報も含まれます）。この情報は、ODBC クライアント・ドライバで使用されます。

以下は、InterSystems ODBC ドライバ用の初期化ファイルのサンプルです。

```
[ODBC Data Sources]
sampleodbc=sampleodbc

[sampleodbc]
Driver          = /usr/irissys/bin/libirisodbc.so
Description     = InterSystems IRIS ODBC driver
Host            = localhost
Namespace       = USER
UID             = _SYSTEM
Password        = SYS
Port            = 51774
Protocol        = TCP
Query Timeout   = 1
Static Cursors  = 0
Trace           = off
TraceFile       = iodbctrace.log
Authentication Method   = 0
Security Level          = 2
```

```
Service Principal Name  = iris/localhost.domain.com

[Default]
Driver = /usr/irissys/bin/libirisodbc.so
```

このファイルには、以下の変数が含まれます。

- ODBC Data Sources ー このセクションには、ファイルで定義されているすべての DSN がリストされます。各エントリは DSNName=SectionHeading という形式で表します。DSNName はクライアント・アプリケーションが指定する名前で、SectionHeading はこのファイルの DSN 情報を表す見出しを指定します。
- Driver ー この DSN に使用するクライアント・ドライバ・ファイルの場所を指定します。この例では、libirisodbc.so ファイルです。
- Description ー DSN の説明を指定します (オプション)。
- Host ー DSN の IP アドレスを指定します。“127.0.0.1” のように、ドットで区切られた 10 進法の数字、あるいは 4 個の数字で表します。
- Namespace ー DSN のネームスペースを指定します。
- UID ー DSN にログインするためのユーザ名を指定します。既定では、これは _SYSTEM です (大文字と小文字は区別されません)。
- Password ー UID エントリにより指定されたアカウントのパスワードを指定します。既定のユーザ名 _SYSTEM の場合、パスワードは SYS です。UID と異なり、パスワードでは大文字と小文字が区別されます。

  注釈　ODBC 基準ではユーザ名とパスワードのクリア・テキストでの保存が許可されるため、サンプルの初期化ファイルには、サンプル DSN へのアクセスに必要なユーザ名とパスワードが含まれます。これは単なる例にすぎません。セキュリティで保護された ODBC プログラムでは、この情報の入力をユーザに要求するプロンプトが表示されるだけで保存はされません。この場合、入力された情報は初期化ファイル内に表示されません。

- Port ー DSN に接続するポートを指定します (″構成パラメータ・ファイル・リファレンス″ の “DefaultPort” を参照)。
- Protocol ー DSN に接続するためのプロトコルを指定します。InterSystems の場合は、常に TCP です。
- Query Timeout ー 1 の場合は、ODBC クライアント・ドライバが ODBC クエリ・タイムアウトの設定値を無視します。

  ODBC クエリ・タイムアウト設定は、特定の動作が完了するまでにクライアントが待機する時間を指定します。動作が指定時間内に完了しなかった場合、自動的にキャンセルされます。ODBC API には、このタイムアウト値をプログラムで設定する機能があります。ただし、一部の ODBC アプリケーションではこの値をハード・コーディングします。ご使用の ODBC アプリケーションでタイムアウト値を設定できず、そのタイムアウト値が小さすぎる場合、クエリ・タイムアウトを無効にするオプションにより、タイムアウト値を無効にできます。
- Static Cursors ー 1 の場合は、InterSystems ODBC クライアント・ドライバのスタティック・カーソル・サポートを有効にします。0 の場合は、ODBC カーソル・ライブラリのカーソル・サポートを使用します。通常、ODBC カーソル・ライブラリを使用しない特別な理由がない限り、このフラグはオフ (0) です。
- Trace ー ドライバ・マネージャでロギングを実行するかどうかを指定します。実行する場合は “オン”、実行しない場合は “オフ” を指定し、既定ではオフです (詳細は、“UNIX® の ODBC ログ” を参照)。
- TraceFile ー Trace エントリによりロギングが有効になっている場合、ドライバ・マネージャのログ・ファイルの場所を指定します。
- Authentication Method ー パスワード認証の場合は 0 を、Kerberos の場合は 1 を指定します。
- Security Level ー 認証に Kerberos を使用する場合に指定します。以下の値を指定できます。
  - 1 = Kerberos
  - 2 = Kerberos とパケット整合性

- 3 = Kerberos と暗号化

- Service Principal Name ― 認証に Kerberos を使用する場合に指定します。InterSystems を表すサービス・プリンシパルの名前を指定する必要があります。

Kerberos の詳細は、"Kerberos 認証" を参照してください。

3.2 odbcinst による DSN の設定

UNIX® の ODBC インストールには、プログラム **odbcinst** が含まれます。場所はインストールに依存しますが、例えば **/usr/local/bin** にある場合があります。

UNIX® インストールには、install-dir**¥dev¥odbc¥redist¥unixodbc** にテンプレート・ファイルが 2 つ含まれます。これらは、以下のとおりです。

- **odbc.ini_unixODBCtemplate** ― サンプル DSN エントリ・テンプレート
- **odbcinst.ini_unixODBCtemplate** ― インターシステムズ・ドライバ・テンプレート

これらのテンプレート・ファイルを構成に合うように編集します。これらを使用するには、以下の方法で **odbcinst** を呼び出すことができます。

- ドライバを登録するには、フラグ `-i -d -f` と **odbcinst.ini** ファイルを指定します。以下に例を示します。

```
odbcinst -i -d -f   odbcinst.ini_unixODBCtemplate
```

- ローカル DSN を追加するには、フラグ `-i -s -h -f` と **odbc.ini** ファイルを指定します。以下に例を示します。

```
odbcinst -i -s -h -f  odbc.ini_unixODBCtemplate
```

- システム DSN を追加するには、フラグ `-i -s -l -f` と **odbc.ini** ファイルを指定します。以下に例を示します。

```
odbcinst -i -s -l -f  odbc.ini_unixODBCtemplate
```

From: **¥dev¥odbc¥redist¥unixodbc¥odbcinst.ini_unixODBCtemplate**

```
[InterSystems ODBC]
UsageCount=1
Driver=/home/iris/bin/libirisodbc.so
Setup=/home/iris/bin/libirisodbc.so
SQLLevel=1
FileUsage=0
DriverODBCVer=02.10
ConnectFunctions=YYN
APILevel=1
DEBUG=1
CPTimeout=<not pooled>
```

3.3 TLS 構成ファイルの設定

InterSystems IRIS では、TLS 構成用のテンプレート・ファイルが 2 つ提供されています。これらのファイルは install-dir**¥dev¥odbc¥redist¥ssl** にあります。このディレクトリには、詳細な情報が記載された **readme.txt** も含まれます。

- **irisodbc.ini.template** ― TLS 接続用に **odbc.ini** ファイルのエントリを構成する方法を示します。

- odbcssl.ini.template ー TLS 構成ファイルの例です。

irisodbc.ini.template

これは、サンプルの odbc.ini ファイルで、TLS 接続を定義する [SampleSSL] というエントリが記述されています。機能するファイルは通常、install-dir/mgr/irisodbc.ini という名前です。

```
[ODBC Data Sources]
SamplesSSL = SampleTLS

[SampleTLS]
Driver = /home/guest/iris/bin/libirisodbc35.so
Description = IRIS ODBC driver
Host = localhost
Namespace = SAMPLES
UID = _SYSTEM
Password = SYS
Port = 51774
Protocol = TCP
Query Timeout = 1
Static Cursors = 0
Trace = off
TraceFile = iodbctrace.log
Service Principal Name = iris/localhost.domain.com

Authentication Method = 2
Security Level = 10
SSL Server Name = SampleSSLConfig
```

上の例では、最後の 3 行で TLS 接続が指定されています。以下のように値を定義する必要があります。

- Authentication Method は 2 に設定する必要があります。
- Security Level は 10 に設定する必要があります。
- SSL Server Name は適切な名前の構成に設定する必要があります。この例では、SampleSSLConfig は、以下のサンプル・ファイル odbcssl.ini で定義されている SSL Server Name です。

odbcssl.ini

これはサンプルの TLS 構成ファイルです。プロセスがこれらの値で TLS 接続を開始するには、以下を行う必要があります。

- このファイルの名前 (<path>/odbcssl.ini) を環境変数 ISC_SSLconfigurations で指定する必要があります。
- プロセスが、(前の例に示されているように) [SampleSSLConfig] を SSL Server Name として指定した DSN を使用している必要があります。

```
[SampleSSLConfig]
CAFile=./CA.cer
CertFile=./Client.cer
KeyFile=./Client.key
Password=MixOfAlphaNumericAndPuncChars!
KeyType=2
Protocols=28
CipherList=ALL:!aNULL:!eNULL:!EXP:!SSLv2
VerifyPeer=1
VerifyDepth=9
```

この例では、以下の値を定義します。

- CAFile ー サーバの証明書を検証するために使用する 1 つ以上の証明書が含まれるファイルを指定します。
- CertFile ー クライアントの証明書が含まれるファイルを指定します。
- KeyFile ー クライアントの秘密鍵ファイルが含まれるファイルを指定します。
- Password ー 該当する場合、クライアントの秘密鍵ファイルのパスワードです。

- KeyType ー クライアントが使用する秘密鍵のタイプを指定します。
  - 1 ー DSA
  - 2 ー RSA (既定)

- Protocols ー クライアントが実行できる TLS のバージョンを指定します。
  - 1 ー SSLv2
  - 2 ー SSLv3
  - 4 ー TLSv1.0
  - 8 ー TLSv1.1
  - 16 ー TLSv1.2

  プロトコルの組み合わせは、個々の番号を加算することによって指定します。例えば、既定の設定は 28 (TLSv1 + TLSv1.1 + TLSv1.2) です。

- CipherList ー 有効な暗号スイートのリストを指定します。
- VerifyPeer ー 相手証明書認証レベルを指定します。
  - 0 ー なし (証明書の検証に失敗しても続行します。)
  - 1 ー 必要 (証明書の検証が成功する場合にのみ続行します。既定値です。)

- VerifyDepth ー 相手証明パスで許可された最大 CA 証明書数を指定します。

これらの値の詳細は、インターシステムズの "TLS ガイド" を参照してください。

3.4 初期化ファイルの名前と場所

初期化ファイルには任意の名前を使用できますが、一般に、ユーザ個人のディレクトリにある場合は .odbc.ini、ODBC 指定のディレクトリにある場合は odbc.ini と呼ばれます。InterSystems のサンプル・ファイルは irisodbc.ini と呼ばれ、install-dir/mgr ディレクトリに格納されています。

このファイルを見つけるため、InterSystems ODBC クライアント・ドライバは iODBC と同じ検索順を使用します。以下の順番どおりに、以下の場所を検索します。

1. ODBCINI 環境変数が定義されている場合、これに指定されているファイル。定義されている場合は、この変数は以下のようにパスとファイルを指定します。

```
ODBCINI=/usr/irissys/irisodbc.ini
export ODBCINI
```

2. $HOME が定義されており .odbc.ini が存在する場合、ユーザの $HOME 変数に指定されたディレクトリの .odbc.ini ファイル。
3. $HOME が定義されていない場合、passwd ファイル内に指定された “home” ディレクトリの .odbc.ini ファイル。

4. システム全体に、SYSODBCINI 環境変数が定義されている場合、これに指定されているファイル。定義されている場合は、この変数は以下のようにパスとファイルを指定します。

```
SYSODBCINI=/usr/irissys/irisodbc.ini
export SYSODBCINI
```

5. odbc.ini ファイルは、iODBC ドライバ・マネージャ (/etc/) を構築する既定のディレクトリにあるため、完全なパス、およびファイル名は /etc/odbc.ini です。

異なる odbc.ini ファイルを使用するには、InterSystems サンプル初期化ファイルを削除するか名前を変更して、ドライバ・マネージャが $HOME または /etc/odbc.ini のパスを検索できるようにします。例えば、install-dir/bin に移動して以下のコマンドを実行します。

```
mv libodbc.so libodbc.so.old
```

その後、ユーザ定義の odbc.ini をドライバ・マネージャが検索可能な etc/odbc に移動します。

4

UNIX® システムでの ODBC のインストールと検証

この章では、UNIX® および関連オペレーティング・システムでの ODBC のインストールと検証に関する詳細な情報を提供します。以下のトピックについて説明します。

- 共有オブジェクトの依存関係に関するトラブルシューティング – 共有オブジェクトの依存関係の確認方法。
- スタンドアロン・インストールの実行 – InterSystems ODBC クライアント・ドライバとサポートされるドライバ・マネージャの UNIX® へのインストール。
- iODBC のカスタム・インストールと構成 – iODBC ドライバ・マネージャのインストールと構成、およびiODBC のための PHP の構成。
- キー・ファイル名 – インストールされる重要なコンポーネントの具体的なファイル名。

4.1 共有オブジェクトの依存関係に関するトラブルシューティング

インストールを実行したら、他の共有オブジェクトについて依存関係を確認し、すべての問題を修正する必要があります。そのプロセスは、以下のとおりです。

1. 適切なコマンドを使用して、InterSystems ODBC ドライバの動的な依存関係をリストします。

   例えば、Solaris やその他のプラットフォームでは、ldd コマンドを使用します。

   ```
   # ldd install-dir/bin/libirisodbc.so
   ```

   install-dir は、InterSystems のインストール・ディレクトリです。依存関係が検出されなかった場合は、以下のようなメッセージが表示されます。

   ```
   libstlport_gcc.so => not found
   ```

2. エラーが発生しない場合は、すべての依存関係は有効です。エラーが発生した場合は、以下のコマンドを実行して、現在のディレクトリ内を見るために共有オブジェクト・ローダを強制します。

   ```
   # sh
   # cd install-dir/bin
   # LD_LIBRARY_PATH=`pwd`:$LD_LIBRARY_PATH
   # export LD_LIBRARY_PATH
   ```

sh コマンドは、Bourne シェルを起動し、cd コマンドは適切なディレクトリに変更し、export コマンドは共有オブジェクトを検索するパスを設定します。

ただし、AIX® では、LD_LIBRARY_PATH の代わりに LIBPATH を使用します。

3. 現在のディレクトリにパスを追加した後、再度 ldd を実行して、欠けている依存関係がないかどうかを確認します。共有オブジェクトがひとつも見つからない場合は、ODBC クライアント・ドライバと同じディレクトリにそれらを追加してください。

4.2 スタンドアロン・インストールの実行

InterSystems の標準インストールでは、既定で ODBC の完全インストールが実行されます。カスタム・インストール (″インストール・ガイド″ で説明) を実行する場合は、“[SQL クライアントのみ]” オプションを選択して、クライアント・アクセス・コンポーネント (ODBC クライアント・ドライバ) のみをインストールできます。

ただし、InterSystems ODBC 用のスタンドアロン・インストーラも提供されています。このインストーラを使用するには、以下の手順に従います。

1. クライアントをインストールするディレクトリ (/usr/irisodbc/ など) を作成します。
2. 作成したディレクトリに、適切な zip 圧縮 tar ファイルをコピーします。

   ./dist/ODBC/ ディレクトリには、以下のような名前の zip 圧縮 tar ファイルが含まれます。

   ```
   ODBC-release-code-platform.tar.gz
   ```

   release-code はリリース固有のコード (InterSystems のバージョンとリリースによって異なります)、platform は ODBC クライアントが実行されるオペレーティング・システムを示します。
3. 作成したディレクトリに移動して、以下のように手動で .tar ファイルをアンパックします。

   ```
   # gunzip ODBC-release-code-platform.tar.gz
   # tar xvf ODBC-release-code-platform.tar
   ```

   bin ディレクトリと dev ディレクトリが作成され、ファイルのセットがインストールされます。
4. 作成したディレクトリにある ODBCInstall プログラムを実行します。このプログラムは、複数のサンプル・スクリプトを作成し、mgr ディレクトリで irisodbc.ini を構成します。以下に例を示します。

   ```
   # pwd
   /usr/irisodbc
   # ./ODBCInstall
   ```

注釈 **正しいプラットフォーム名の識別**

一部のリリースでは、./dist/ODBC/ ディレクトリに、必要なファイルを識別するプラットフォーム名を表示するために以下のコマンドが含まれます。

```
# ./cplatname identify
```

このコマンドは、このコマンドが必要ないリリースには存在しません。

4.2.1 UNIX® および関連プラットフォーム向け SQL ゲートウェイ・ドライバ

<install-dir>/bin/ ディレクトリには、SQL ゲートウェイによって使用される以下のバージョンの共有オブジェクトが格納されています。これにより、InterSystems IRIS から他の ODBC クライアント・サーバへの接続が可能になります。スタンドアロン・インストールでは、これらのファイルはインストールされません。

iODBC にリンクされている

- **cgate.so** — 8 ビット ODBC をサポートします。
- **cgateiw.so** — Unicode ODBC をサポートします。

unixODBC にリンクされている

- **cgateu.so** — 8 ビット ODBC をサポートします。
- **cgateur64.so** — 64 ビット unixODBC 用に 8 ビット ODBC をサポートします。

詳細は、“ODBC データ・ソースとしての InterSystems データベースの使用法 (UNIX®)” を参照してください。

注釈 **UNIX® システムでの共有ライブラリ・パスの設定**

UNIX® システムでサードパーティ製共有ライブラリを使用する場合は、InterSystems IRIS LibPath パラメータを設定して **LD_LIBRARY_PATH** を定義する必要があります (″構成パラメータ・ファイル・リファレンス″ の “LibPath” を参照してください)。これは、特権を持たないユーザがパスを変更できないようにするためのセキュリティ対策です。

4.3 iODBC のカスタム・インストールと構成

カスタム化された条件のもとで運用するために、独自の iODBC ドライバ・マネージャを構築することが可能です。iODBC 実行可能ファイルおよびインクルード・ファイルは、install-dir/**dev/odbc/redist/iodbc/** ディレクトリにあります。これらのディレクトリを使用してアプリケーションを構築するには、LD_LIBRARY_PATH (AIX® では LIBPATH) とインクルード・パスを設定する必要があります。

iODBC ドライバ・マネージャをカスタマイズすることもできます。iODBC のウエブ・サイト (www.iodbc.org) からソースをダウンロードして、表示される指示に従います。

4.3.1 iODBC を使用した PHP の構成

InterSystems の ODBC 機能は PHP (PHP : Hypertext Processor) と組み合わせて使用できます。PHP は、動的に生成されるページを作成するスクリプト言語です。そのプロセスは、以下のとおりです。

1. インストールを実行するマシンで、root 特権を取得します。
2. iODBC ドライバ・マネージャをインストールします。以下はその方法です。
   a. キットをダウンロードします。
   b. この章で前述したように、標準インストールと構成を行います。
   c. iODBC の Web サイト (www.iodbc.org) の iODBC+PHP HOWTO ドキュメントの説明に従って、ドライバ・マネージャを PHP で使用するように構成します。

既定の PHP 構成のセキュリティ保護のため、iODBC PHP の例では、LD_LIBRARY_PATH (AIX® では LIBPATH) はセットされていない点に注意してください。また、LD_LIBRARY_PATH を使用しないでこれを登録するには、libiodbc.so を /usr/lib にコピーし、ldconfig を実行します。

3. PHP ソース・キットを https://www.php.net からダウンロードし、解凍します。
4. Apache HTTP サーバ・ソース・キットを http://httpd.apache.org/ からダウンロードし、解凍します。
5. PHP を構築し、インストールします。
6. Apache HTTP サーバを構築し、インストール後、起動します。
7. Apache ルート・ディレクトリの info.php を使用して PHP と Web サーバをテストします。Apache ルート・ディレクトリは、Apache 構成ファイルで指定されています (通常 httpd.conf)。この URL は、http://127.0.0.1/info.php です。
8. $HOME 環境変数が定義されていない場合は、InterSystems 特有の初期化ファイル irisodbc.ini を /etc/odbc.ini にコピーします。Apache Web サーバでは、この場所を指定した方が好結果が得られます。
9. libirisodbc.so クライアント・ドライバ・ファイルを構成し、テストします。
10. InterSystems ODBC キットから sample.php ファイルを Apache ルート・ディレクトリ (info.php が存在するディレクトリ) にコピーし、その場所を、使用しているマシンでの InterSystems インストール・ディレクトリの場所に合わせて調整します。
11. その後、ブラウザで http://127.0.0.1/sample.php を指して、SAMPLES ネームスペースを使用する sample.php プログラムを実行します。

4.4 キー・ファイル名

構成のニーズによっては、インストールされているコンポーネントの特定のファイル名を知っておくと便利な場合があります。次のリストでは、install-dir は InterSystems インストール・ディレクトリ (ご使用のシステムで `$SYSTEM.Util.InstallDirectory()` が返すパス) です。

ODBC ドライバ・マネージャ

install-dir/bin/ ディレクトリには、以下のドライバ・マネージャが格納されています。

- libiodbc.so ― iODBC ドライバ・マネージャです。8 ビット ODBC API と Unicode ODBC API の両方をサポートします。
- libodbc.so ― unixODBC ドライバ・マネージャです (ODBC 8 ビット API で使用)。

注釈 **64 ビット UNIX® プラットフォーム上の ODBC**

ODBC 仕様のリリース間で、SQLLen や SQLULen などのさまざまなデータ型が 32 ビット値から 64 ビット値へと変わりました。これらの値は、iODBC ではこれまでも常に 64 ビットでしたが、unixODBC では 32 ビットから 64 ビットに変わりました。unixODBC バージョン 2.2.14 以降、既定のビルドでは 64 ビットの整数値が使用されます。InterSystems ドライバは、32 ビット unixODBC と 64 ビット unixODBC の両方で使用できます。

InterSystems ODBC クライアント・ドライバ

InterSystems ODBC クライアント・ドライバは、ODBC 2.5 と ODBC 3.5 の両方に向けて用意されています。ODBC 3.5 バージョンは、3.5 の要求を従来の 2.5 に自動的に変換するため、ほとんどの場合どちらのドライバも使用できます。install-dir/bin/ ディレクトリには、以下のバージョン (*.so または *.sl) が格納されています。

iODBC 準拠のドライバ

- libirisodbc ― 8 ビット ODBC 2.5 の既定のドライバです。

- **libirisodbc35** ― 8 ビット ODBC 3.5 をサポートします。
- **libirisodbciw** ― Unicode ODBC 2.5 をサポートします。
- **libirisodbciw35** ― Unicode ODBC 3.5 をサポートします。
- **libirisodbciw.dylib** ― MAC OS 用の Unicode ODBC をサポートします。

unixODBC 準拠のドライバ

- **libirisodbcu.** ― 8 ビット ODBC 2.5 の既定のドライバです。
- **libirisodbcu35** ― 8 ビット ODBC 3.5 をサポートします。
- **libirisodbcur64** ― 64 ビット unixODBC 用に 8 ビット ODBC 2.5 をサポートします。
- **libirisodbcur6435** ― 64 ビット unixODBC 用に 8 ビット ODBC 3.5 をサポートします。

その他のファイル

install-dir/**mgr/irisodbc.ini** ファイルは、サンプルの ODBC 初期化ファイルです。

テスト・プログラム用ファイルについては、“InterSystems ODBC 構成のテスト”に説明があります。

5

Python および Node.js 用の ODBC サポート

ここで説明するオープン・ソース・モジュールは、Python および Node.js 用の言語固有の ODBC インタフェースを提供します。

- pyodbc Python ODBC ブリッジのサポート ー **pyodbc** は Python DB API 2.0 仕様を実装します。
- Node.js リレーショナル・アクセスのサポート ー **node-odbc** は、Node.js クライアント・アプリケーションの ODBC データベース・アクセスを可能にします。

5.1 pyodbc Python ODBC ブリッジのサポート

pyodbc は、DB API 2.0 仕様 (PEP 249 -- Python Database API Specification v2.0) を実装し、ODBC を活用して基盤となるデータベースにアクセスするオープン・ソース Python モジュールです。インターシステムズでは、リレーショナル・パラダイムを使用して Python からデータベースにアクセスする手段として、**pyodbc** の使用をサポートしています。このモジュールは、前のバージョンの InterSystems IRIS でも使用できます。一般情報は、pyodbc の GitHub サイトを参照してください。

注釈　DB API 2.0 のインターシステムズ固有の実装が InterSystems IRIS 2022.1 で導入されており (″Native SDK for Python の使用法″ の “Python DB-API の使用法” を参照)、これをすべての新しい Python 開発で使用することが推奨されています。**pyodbc** 実装は引き続きサポートされ、古いバージョンの InterSystems IRIS に接続するクライアント・アプリケーションで必要となる場合があります。

5.1.1 インストール

Windows 向けおよび Linux および関連オペレーティング・システム向けのインストール情報が掲載されているサイトがいくつかあります。

- pyodbc の GitHub サイト : pyodbc Python ODBC bridge
- pyodbc Wiki : Wiki Home
- Microsoft 製品への pyodbc のインストール : Python SQL Driver - pyodbc
- 一般的な Python のドキュメント : Python

インストール手順はシンプルです。

- Python ダウンロード・ページから Python 2 または 3 (Unicode をサポート) をインストールします。

- パスに Python を含むコンソールから、以下を実行します。

```
pip install pyodbc
```

5.1.2 macOS X へのインストール

macOS X へのインストールは、UNIX® プラットフォームとほぼ同じです (″Python Releases for Mac OS X″ を参照)。

- homebrew をインストールします。
- unixODBC をインストールします。
- pip install を実行します。

```
pip install --upgrade --global-option=build_ext
--global-option="-I/usr/local/include"  --global-option="-L/usr/local/lib"
```

5.1.3 テスト・プログラム

以下のテスト・プログラムは、pyodbc を使用して InterSystems IRIS データベースにアクセスする方法を示しています。InterSystems ODBC ドライバでサポートされる接続キーワードを示した例については、“ODBC 初期化ファイルの構造”を参照してください。

test.py

```
import pyodbc
import time

input("Hit any key to start")

dsn = 'IRIS Samples'
server = '127.0.0.1'
database = 'USER'
username = '_SYSTEM'
password = 'SYS'
#cnxn = pyodbc.connect('DRIVER={InterSystems ODBC35};SERVER='+server+';
#   PORT='+port+'; DATABASE='+database+';UID='+username+';PWD='+ password)
cnxn = pyodbc.connect('DSN='+dsn+';')
lowptr=cnxn.getinfo(127)
highptr=cnxn.getinfo(136)
#value = PyLong_FromUnsignedLongLong(lowptr)
#print("%#5.8x"% (value))

print ("Connection high pointer: ")
print (format(highptr, '02x'))
print ("Connection high pointer: ")
print("%#5.8x"% (highptr))
print ("Connection low pointer: ")
print("%#5.8x"% (lowptr))
cursor = cnxn.cursor()
start= time.perf_counter()


#Sample select query
cursor.execute("SELECT * from Sample.Person")
row = cursor.fetchone()
#while row:
#    print(row)
#    row = cursor.fetchone()

end= time.perf_counter()
print ("Total elapsed time: ")
print (end-start)
input("Hit any key to end")
```

以下のように変更することで、Unicode データが返されなくなり、UTF-8 データが直接返されます。

```
cnxn.setdecoding(pyodbc.SQL_CHAR, encoding='latin1')
cnxn.setencoding(str, encoding='latin1')
```

ここでは、対象を絞ったドライバを使用することで、ドライバ・マネージャが UCS-2 または UCS-4 の Unicode を使用するのを回避すると共に、特定のドライバ・マネージャの構築方法に合ったドライバを提供することによる複雑さを回避します。他の Unicode オプションについては、pyodbc Wiki の Unicode のセクションを参照してください。

5.2 Node.js リレーショナル・アクセスのサポート

node-odbc オープン・ソース Node.js モジュールは、Node.js クライアント・アプリケーションの ODBC データベース・アクセスを可能にします。node-odbc のサイト (https://github.com/wankdanker/node-odbc) によると、このモジュールは“unixODBC とそのサポート対象ドライバに対する node.js 用の非同期/同期インタフェース”ですが、Windows でも Windows ドライバ・マネージャと連携して機能します。InterSystems IRIS は、どちらのプラットフォームでも **node-odbc** をサポートしています。

5.2.1 依存関係

- InterSystems ODBC ドライバ

  このドライバは、InterSystems IRIS のインストール時に既定でインストールされます。

- Node.js および npm

  Node.js バージョン 8 以降がインストールされていることを確認してください。npm は通常、Node.js と共にインストールされます。

- node-odbc

  node-odbc パッケージは npm を使用して入手することも、Github からローカルにインストールすることもできます。詳細は、Github の node-odbc のサイト (https://github.com/wankdanker/node-odbc) を参照してください。

  node-odbc を構築するには、以下のパッケージが必要です。

  - node-gyp

    node-odbc はソースとして提供されており、node-gyp を使用して npm コマンドで構築します。npm を使用して node-odbc をインストールすると、node-gyp もインストールされることがあります。インストールされない場合は、インストール方法の詳細を node-gyp のサイト (https://www.npmjs.com/package/node-gyp) で参照してください。

    OS または Linux ディストリビューションによっては、node-gyp で node-odbc モジュールを構築するために必要な開発ツールのインストールが必要になることがあります。ここでは、必要なツールやそのインストール方法については説明しません。詳細は、node-gyp および node-odbc のインストール手順を参照してください。

  - Python および関連する開発ツール

    Python は node-gyp の必要条件です。本書の執筆時点では、node-gyp は Python 2.7 に依存していますが、これは、将来 node-gyp の新バージョンが利用可能になったら変わる可能性があります。

- unixODBC (Linux/UNIX のみ)

  Linux で node-odbc を使用するには unixODBC ドライバ・マネージャが必要であり、これはほとんどの Linux ディストリビューションで標準で提供されています。システムにまだインストールされていない場合は、お使いのディストリビューションのインストール手順を参照してください。unixODBC のサイト (http://www.unixodbc.org/) からダウンロードすることもできます。

5.2.2 インストールとセットアップ

- すべての依存関係を確実にインストールします。
  - Node.js と npm (https://nodejs.org/en/download/) ー Node.js バージョン 8 以降がインストールされていることを確認してください。npm も必要であり、通常は Node.js と共にインストールされます。ノード・モジュールを npm を使用してローカルにインストールするか、グローバルにインストールするかを決定します。ローカル・インストールの最初のステップは、プロジェクト・フォルダを定義し、そのフォルダに移動して `'npm init'` を実行することです (以下のセクションの例を参照)。
  - node-gyp (https://www.npmjs.com/package/node-gyp) ー node-odbc を構築するには、このパッケージが必要です。node-gyp はグローバルにインストールするのが適切ですが、ローカル・インストールでも機能します。どちらのケースでも、node-gyp には Python 2.7 も必要です。
  - node-odbc (https://github.com/wankdanker/node-odbc) ー リンク・ページに記載されている手順を使用して、システムにインストールします。これは、IRIS ODBC 用に再構築する必要があるため、多くの場合はローカルにインストールする必要があります。
- UNICODE サポートを削除して、node-odbc を再構築します。./node_modules/odbc/binding.gyp を編集して、`'defines'` 配列の `'UNICODE'` をコメント・アウトします。変更された binding.gyp を保存して、プロジェクト・フォルダで `'npm rebuild'` を実行します。
- 適切な InterSystems ODBC DSN が定義されていることを確認します。Windows では、データ ソース アドミニストレーターを使用できます (“Windows での ODBC データ・ソースの定義” を参照)。Windows 以外のプラットフォームでは、ODBCINI 環境変数を必要な odbc.ini ファイルの場所に定義します (“UNIX® での ODBC データ・ソースの定義” を参照)。これは、node-odbc モジュールをロードする前に JavaScript で定義することもできます。

5.2.3 Ubuntu でのインストールとセットアップの例

このサンプルでは、システムに Node.js と npm がインストール済みであると想定しています。npm を使用して node-odbc をインストールすると、node-gyp もインストールされることがあります。node-odbc を構築する前に、node-gyp モジュールとその依存関係がインストールされている必要があります。

npm init でのプロジェクト・フォルダのセットアップ

`npm init` の各種オプションでは既定値を使用してかまいません。

```
~$ mkdir my_odbc
~$ cd my_odbc
~/my_odbc$ npm init

This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and   save it as a dependency in the
package.json file.

Press ^C at any time to quit.
package name: (my_odbc)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/your_home/my_odbc/package.json:

{
"name": "my_odbc",
"version": "1.0.0",
```

```
"description": "",
"main": "index.js",
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
},
"author": "",
"license": "ISC"
}

Is this OK? (yes)
~/my_odbc$
```

node-gyp のインストール（まだグローバルにインストールされていない場合）

```
~/my_odbc$ npm ls node-gypnpm ls node-gyp
my_odbc@1.0.0 /home/your_home/my_odbc
    (empty)

~/my_odbc$ npm install node-gyp --save

npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN my_odbc@1.0.0 No description
npm WARN my_odbc@1.0.0 No repository field.

+ node-gyp@3.8.0
added 97 packages from 67 contributors and audited 183 packages in 6.749s
found 0 vulnerabilities

~/my_odbc$
```

node-odbc のインストール

```
/my_odbc$ npm install odbc

> odbc@1.4.5 install /home/your_home/my_odbc/node_modules/odbc
> node-gyp configure build

make: Entering directory '/home/your_home/my_odbc/node_modules/odbc/build'
CXX(target) Release/obj.target/odbc_bindings/src/dynodbc.o
SOLINK_MODULE(target) Release/obj.target/odbc_bindings.node
COPY Release/odbc_bindings.node
make: Leaving directory '/home/your_home/my_odbc/node_modules/odbc/build'
npm WARN my_odbc@1.0.0 No description
npm WARN my_odbc@1.0.0 No repository field.

+ odbc@1.4.5
added 4 packages from 10 contributors and audited 187 packages in 6.187s
found 0 vulnerabilities

~/my_odbc$
```

上記によって警告が大量に生成されることがありますが、パッケージが正常に追加されている限り、無視してかまいません。

UNICODE サポートの削除および node-odbc の再構築

./node_modules/odbc/binding.gyp を編集して、UNICODE サポートを削除します。

```
$/my_odbc$ nano ./node_modules/odbc/binding.gyp
```

定義セクションが以下のようになっていることを確認します。

```
'defines' : [
# 'UNICODE'
],
```

次に、node-odbc を再構築します。

```
~/my_odbc$ npm rebuild
```

このコマンドによって警告が多数生成されることがありますが、無視してかまいません。エラーが発生しておらず、新しいモジュールが正常にリンクされたことを確認します。

サンプル・プログラムのセットアップ

以下の JavaScript コードを使用して、ODBC 接続をテストできます。この手順では、InterSystems IRIS サーバが実行されていて、DSN が適切に定義されている必要があります。

```
// update this line to reference the location of irisodbc.ini on your system
  process.env.ODBCINI = process.env.ODBCINI || '/opt/isc/iris/inat/mgr/irisodbc.ini';

  var db = require("odbc")();

  let cn = 'DSN=';

  if (process.platform == "win32") {
    // Windows
    cn += 'Sampleodbc;';
  } else if (process.platform == "darwin") {
    // Mac OS
    cn += 'Userunixodbc;';
  } else if (process.platform == "linux") {
    cn += 'Userunixodbc;';
  }
  console.log(cn);

  db.open(cn, function (err) {
    if (err) {
      return console.log(err);
    }
    console.log('I am connected')    ;

    db.query('select * from sample.person where id<3',  function cb(err, data) {
      if (err) {
        console.error(err);
      } else {
        console.log(data);
      }
    });
    db.close(function () { });
  });
```

このコードは、DSN で指定されたネームスペースで Sample.Person クラスが定義およびコンパイルされ、3 未満の ID 値を持つデータがあると想定しています。

サンプルの実行

```
  ~/my_odbc$ node index.js
  ~/my_odbc$ node sample.js

  DSN=Userunixodbc;
  I am connected
  [ { ID: 1,
    Age: 3,
    DOB: '2015-09-28',
    FavoriteColors: '',
    Name: 'Ulman,George L.',
    SSN: '293-31-5406',
    Spouse: 0,
    Home_City: 'Newton',
    Home_State: 'MI',
    Home_Street: '6958 Main Avenue',
    Home_Zip: '20649',
    Office_City: 'Xavier',
    Office_State: 'NY',
    Office_Street: '7313 Madison Avenue',
    Office_Zip: '73226' },
  { ID: 2,
    Age: 16,
    DOB: '2002-04-07',
    FavoriteColors: 'Green',
    Name: 'Pascal,Vincent A.',
    SSN: '973-94-3185',
    Spouse: 0,
    Home_City: 'Xavier',
    Home_State: 'ND',
    Home_Street: '3788 Madison Drive',
    Home_Zip: '80569',
    Office_City: 'Washington',
    Office_State: 'SC',
    Office_Street: '1206 Second Place',
```

```
  Office_Zip: '37389' } ]
~/my_odbc$
```

6

ログと環境変数

この章では、トラブルシューティングを実行するために使用できるツールについて説明します。

注意　　ログは、トラブルシューティングを実行する必要がある場合のみ有効にします。ログを有効にするとパフォーマンスが大幅に低下するため、通常の操作時は有効にしないでください。

6.1 Windows の ODBC ログ

Windows で ODBC データ・ソースのログを有効にするには、[ODBC データ ソース アドミニストレーター] を使用して、DSN のログ情報を変更します (使用法の詳細は、“ODBC データ ソース アドミニストレーターによる DSN の作成”を参照してください)。[ODBC データ ソース アドミニストレーター] で、以下の変更を行います。

- クライアント・ドライバのログを有効にするには、ログに記録したい DSN の定義を見つけて開き、[ODBC　　] (または [　] や同様の表現) というラベルのボックスにチェックを付けます。
- ドライバ・マネージャのログを有効にするには、[　　] タブをクリックして、[　　] ボタンをクリックします。[　　] フィールドでは、トレース・ファイルの名前と場所を指定します。

注釈　　既定のログ・ファイル名は IRISODBC.log で、既定の場所は C:¥Users¥Public¥Logs です。IRISODBCTRACEFILE 環境変数を設定することにより、これらの値を変更できます (この章で後述する“ODBC 環境変数”を参照してください)。

6.2 UNIX® の ODBC ログ

UNIX® では、以下のように ODBC のログを有効にします。

- クライアント・ドライバのログを有効にするには、IRISODBCTRACE 環境変数を使用します (後述する“ODBC 環境変数”の章を参照してください)。さらに ODBC 初期化ファイルも構成します。
- ドライバ・マネージャのログを有効にするには、ODBC 初期化ファイルで Trace エントリを設定します (“ODBC データ・ソースとしての InterSystems データベースの使用法 (UNIX®)”の“ODBC 初期化ファイルの構造”を参照してください)。同ファイル内の TraceFile エントリは、作成するログ・ファイルの名前を指定します。

Tipヒント ログを有効にしてもログ・ファイルが更新されない場合は、ファイルに対する書き込み権限がないか、ログを有効にする前に SO にクライアント・アプリケーションがロードされていたかのいずれかです。後者の場合、クライアント・アプリケーションを停止して再起動し、SO の再ロードおよびログ・フラグの取得をアプリケーションに強制します。

6.3 ODBC 環境変数

このセクションでは、InterSystems ODBC クライアント・ドライバを制御する環境変数につて説明します。通常、これらの環境変数は、デバッグまたは各種診断を目的としてのみ使用します。

IRISODBCDEFTIMEOUT

この変数により、ログインの既定のタイムアウトの有効時間を指定できます。この変数の単位は秒です。

IRISODBCPID

このブーリアン変数により、ログ・ファイル名にプロセス ID 番号を自動的に付けるかどうかを制御します。番号を付ける場合は値を `1` に設定し、付けない場合は `0` に設定します。既定では、番号は付きません。

IRISODBCPID が有効であり、ベースのログ・ファイルが **IRISODBC.log** で、現在使用しているディレクトリに属している場合、ID が 21933 のプロセスは、**IRISODBC.log.21933** のログ・ファイルを生成します。

IRISODBCPID と IRISODBCTRACEFILE は共にファイル名に影響します。例えば Windows で IRISODBCTRACEFILE を使用してログ・ファイルのベース・ファイル名を (**C:/home/mylogs/mylog.txt** などに) 設定して IRISODBCPID を有効にすると、ログ・ファイルの名前は **C:/home/mylogs/mylog.txt.21965** のような形式になります。

IRISODBCTRACE (UNIX® のみ)

このブーリアン変数は、クライアント・ドライバのロギングを制御します。クライアント・ドライバのロギングを有効にする場合は値を `1` に設定し、無効にする場合は `0` に設定します。詳細は、この章で前述した “UNIX® の ODBC ログ” を参照してください。

IRISODBCTRACEFILE

この変数は、ログ・ファイルの位置と名前を指定します。これは、一意のディレクトリにログ・ファイルを置き、このファイルに一意の名前を与えるときに便利です。ログ・ファイルの既定名は **IRISODBC.log** です。既定の場所は、以下のとおりです。

- UNIX® の場合、ログは既定で現在のディレクトリに生成されます。
- Windows の場合、ログ・ファイルの既定の場所は **%PUBLIC%¥Logs** です。このディレクトリにはすべてのユーザがアクセスでき、ログの作成場所を 1 か所に集約できるメリットがあります。

IRISODBCTRACETHREADS

この変数はログにスレッド情報を含めるかどうかを制御します。スレッド情報を含める場合は値を `1` に設定し、含めない場合は `0` に設定します。

スレッド化されたアプリケーションをデバッグする必要がある場合は、このような追加のロギングを有効にすると便利な場合があります。ただし、ほとんどの ODBC アプリケーションでは、多くの余分な行がログに追加されます。