Informatica

Informatica® Application Integration
Winter 2019

Amazon S3 コネクタガイド

Informatica Application Integration Amazon S3 コネクタガイド
Winter 2019
2019 年 3 月

発行日: 2020-01-27

目次

序文

アプリケーションの統合向けの Amazon S3 コネクタガイドには、Amazon Simple Storage Service（S3）を使用して接続を設定および使用する方法に関する情報が記載されています。このガイドは、Amazon S3 環境を理解し、アプリケーションの統合を使用して接続を作成する方法を理解していることを前提としています。

第 1 章

Amazon S3 コネクタについて

この章では、以下の項目について説明します。

Amazon S3 コネクタ

アプリケーションの統合は、メッセージやファイルのイベントベースの処理のために使用できる、いくつかのタイプのリスナベースのコネクタをサポートします。これらのコネクタによって、リスナベースの接続を設定するために使用できる、さまざまな接続属性が公開されます。

Amazon Simple Storage Service（S3）コネクタをファイルモニタとして設定できます。これにより、データオブジェクト、サービス呼び出し、およびイベントを Process Designer で使用できるようになります。

Amazon S3 コネクタを使用すると、ファイルコネクタを使用してローカルファイルを扱うのと同じように、S3 ストレージシステム内のファイルを処理できます。例えば、以下の操作が可能です。

1. S3 バケットで新規オブジェクトを監視して、プロセス用のイベントを生成します。プレーンテキスト、バイナリデータ、添付ファイルとして、または解析済み XML、JSON、区切りコンテンツから構築されたプロセスオブジェクトのセットとして、S3 オブジェクトのメタデータや S3 オブジェクトのコンテンツにアクセスできます。
2. 区切りコンテンツ、任意の文字列データ、バイナリデータ、添付ファイル、またはプロセスオブジェクトのリスト（XML または JSON として表す）として、S3 オブジェクトを S3 バケットに書き込みます。

Amazon S3 オブジェクトストア

Amazon S3 は、キーベースのオブジェクトストアです。ユーザーが割り当てる一意のオブジェクトキーを、後でデータを取得するために使用できます。各オブジェクトは、一意のキーを使用して保存および取得されます。

各アカウントには、データを保存し、Amazon S3 名前空間を編成するための 1 つ以上のバケットが含まれます。オブジェクトは、キー（一意の識別子）とバージョン ID によって、バケット内で一意に識別されます。

次に例を挙げます。

```
http://mybucket.s3.amazonaws.com/myfiles/photo1.jpg
```

ここで、バケットは"mybucket"で、キーは"myFiles/photo1.jpg"です。

S3 の各オブジェクトには、データとメタデータのコンポーネントがあります。メタデータは、名前と値のペアや、最終変更日およびその他の標準 HTTP メタデータ（Content-Type など）を使用してオブジェクトを表します。また、バケットでは、Amazon S3 がバケットを保存する地理的な領域も指定できます。

S3 の詳細については、Amazon の製品情報
（http://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html など）を参照してください。

S3 コネクタの実装

S3 コネクタは、Process Designer を使用して接続オブジェクト内に定義できる、設定可能な開始イベント（イベントソース）とイベントサービス（イベントターゲット）を提供します。

イベントソース（開始イベント）とイベントターゲット（イベントサービスの定義）は共通の属性を持ちます。そのため、1 つのアクセスキー/秘密鍵のペア（S3 アカウント）にすべてのバケットが関連していれば、1 つの S3 接続で、複数の異なるバケットで新規オブジェクトを監視し、他のバケットから区切りコンテンツを読み取り、別のバケットに書き込むことができます。

この柔軟な実装により、関連する一連のタスクを 1 つの接続に統合したり、作業をいくつかの異なる接続に分けることもできます。

イベントソースでの S3 オブジェクトの処理

S3 コネクタがバケット内の S3 オブジェクトを処理すると、それらのオブジェクトが再び処理されることがないようにするため、オブジェクトは自動的に削除されます。その結果、コネクタで使用される S3 バケットは、一時ストレージとして、または分離されたアプリケーションと Secure Agent 間のデータ交換バッファとして定義する必要があります。オブジェクトは、ここから処理して別の場所に移動できます。

コネクタは、各オブジェクトを次のように処理します。

- 定義する開始イベント（イベントソース）は、S3 オブジェクトをバケットから読み取り、それをコンテンツを処理するプロセスに送信します。
- コンテンツが処理されると、S3 オブジェクトはバケットから削除され、イベントソースは次のオブジェクトを読み取ります。
- オブジェクトの処理中に問題が発生した場合、処理は終了され、ログファイルにエラーメッセージが出力されます。次の場合にエラーが発生する可能性があります。
  - S3 オブジェクトが処理中にエラーを起こし、そのコンテンツが処理されなかったために、オブジェクトを削除できない場合。
  - S3 オブジェクトをローカルフォルダに移動できず、予期しない結果が生じる可能性があるためにバケットから削除された場合。例えば、プロセスが実行されている接続の設定にエラーがある場合などです。
  - イベントソースの設定に、誤った暗号化アルゴリズムやパスワードが設定されている場合。
  - 接続が、暗号化された S3 オブジェクトを復号化せずに読み取ろうとしている場合。これが許可されないのは、暗号化キーが元のオブジェクトと一緒にバケットから削除される可能性があるためです。

暗号化アルゴリズム

実装は、S3 クライアントサイド暗号化に基づいています。S3 コネクタは、いくつかの異なる暗号化アルゴリズムをサポートします。暗号化設定は、イベントソースの暗号化とイベントターゲットの復号化の両方に使用されます。暗号化された S3 オブジェクトを読み取るには、オブジェクトを復号化する必要があります。

ユーザーパスワードの文字列に基づいて接続が生成したキーの場合、暗号化キーは、固定ソルト"Informatica" を使用した、繰り返し 4096 の PBKDF2WithHmacSHA1 キー導出アルゴリズムに基づきます。

ユーザーが提供する暗号化キーの場合、キーの長さが正しければ、すべてのキーを使用できます。
以降に説明する、クライアントサイド暗号化設定についても参照してください。

Java Cryptography Extension（JCE）

暗号化アルゴリズムを使用するためには、Java Cryptography Extension（JCE）のインストールが必要になる場合があります。JCE は Oracle の Web サイトからダウンロードできます。次のようにインストールします。

1. ダウンロードした JCE の zip ファイルからファイルを抽出します。
2. 次の jar ファイルを *$JAVA_HOME/jre/lib/security* にコピーします。
   local_policy.jar

   US_export_policy.jar

   注: 指定したディレクトリにこれらの jar ファイルがすでにある場合は、バックアップコピーを作成してから、上書きします。
3. 新しい jar ファイルの使用を開始するために、Secure Agent を再起動します。

これらの手順を実行すると、S3 コネクタがサポートするすべての暗号化アルゴリズムを使用できるようになります。

Amazon S3 イベントタイプ

S3 接続の作成後、その接続と関連付ける 1 つまたは複数のイベントソースおよびイベントターゲットを定義できます。
次のイベントソースタイプを Amazon S3 に使用できます。

- モニタ
- 区切りコンテンツパーサー
- 固定幅コンテンツパーサー

次のイベントターゲットタイプを Amazon S3 に使用できます。

- ライタ
- 区切りコンテンツライタ

第 2 章

Amazon S3 接続

この章では、以下の項目について説明します。

基本的な接続プロパティ

次の表に、［接続］ページの**［プロパティ］**タブで使用可能な基本プロパティを示します。

プロパティ	説明
名前	必須。Process Designer での識別に使用される、この接続の一意の名前。
場所	接続を配置するプロジェクトまたはフォルダの場所。接続の場所を選択するには、適切なプロジェクトまたはフォルダを参照するか、デフォルトの場所を使用します。
説明	オプション。接続の説明。
タイプ	必須。この接続に使用するコネクタまたはサービスコネクタ。設定するタイプを選択します。
実行日時	必須。この接続を実行する Cloud Server または Secure Agent。
接続テスト	直前の接続テストが表示されます（接続タイプでサポートされている場合）。
OData 対応	（接続タイプでサポートされている場合）。OData フィードを有効にし、許可されているユーザーとグループを指定するには、［はい］を選択します。
OData で許可されたロール	オプション。設計時に接続へのアクセス権を持つロール。カスタムロールまたはシステム定義ロールを入力できます。このフィールドには複数のロールを入力できます。

これらの基本プロパティに加え、コネクタによっては次を定義します。

- 接続タイプに適したプロパティ。
- **［イベントソース］**および**［イベントターゲット］**（当てはまる場合）。

［メタデータ］には、接続のパブリッシュ時に生成されたプロセスオブジェクトが表示されます。

Amazon S3 接続のプロパティ

Amazon S3 接続プロパティは、その他のタイプの接続のプロパティと類似しています。

次の表に、接続内のすべてのイベントソースおよびイベントターゲットに適用される、AWS 署名設定該当設定について説明します。

AWS **署名設定**	**説明**
アクセスキー ID	必須。要求者の AWS アクセスキー ID。
シークレットアクセスキー	必須。要求者の AWS シークレットアクセスキー。

次の表に、接続内のすべてのイベントソースおよびイベントターゲットに適用される、Amazon S3 ポリシーとリージョン設定について説明します。

Amazon S3 **設定**	**説明**
ポリシー	プロセスが作成する新規バケットに適用されるバケットポリシー（Amazon で必要な JSON ベースのアクセスポリシー言語を使用）。
Amazon S3 エンドポイント	このプロセスで新規バケットが作成される場合に S3 バケットが存在するリージョン固有の Web サイトエンドポイント。以下に例を示します。 `s3-website-us-east-1.amazonaws.com`
リージョン	新規バケットが存在するリージョン。以下に例を示します。 `us-east1`

リージョンの一部は、署名バージョン 2 と署名バージョン 4 認証をサポートする Amazon S3 バケットをサポートしています。デフォルトでは、接続は署名バージョン 2 認証を使用します。
署名バージョン 4 認証が必要な Amazon S3 バケットとの間でデータを読み書きするマッピングを実行するためには、ターゲットリージョンを **[Amazon S3 エンドポイント]** フィールドで明示的に指定する必要があります。そうしないと、接続は署名バージョン 2 認証を使用し、マッピングに失敗します。Amazon S3 バケットリージョンのサポートの詳細については、Amazon S3 のマニュアルを参照してください。

次の表に、接続内のすべてのイベントソースおよびイベントターゲットに適用される、クライアントサイド暗号化設定について説明します。

クライアントサイド暗号化設定	**説明**
データ暗号化の使用	[はい] の場合、S3 オブジェクトのコンテンツはイベントソースおよびターゲットによって、復号化および暗号化されます。
暗号化アルゴリズム	S3 オブジェクトの復号化および暗号化に使用する暗号化アルゴリズム。

クライアントサイド暗号化設定	説明
暗号化キー	暗号化に使用する Base64 エンコード秘密鍵。鍵の長さは選択した暗号化アルゴリズムに必要な長さ（AES - 128、192、256 ビット; DES - 64 ビット; 3DES - 192 ビット）に一致する必要があります。
暗号化パスワード	暗号化キーの生成に使用する暗号化パスワード。暗号化キーを入力するか、このパラメータを使用して生成します。両方のパラメータが入力された場合は、暗号化キーが使用されます。

第 3 章

Amazon S3 イベントソース

この章では、以下の項目について説明します。

基本的なイベントソースのプロパティ

設定する各接続について、1 つまたは複数のイベントソースを追加できます。イベントソースは、指定した場所の新しいファイルまたはメッセージをリスンまたは監視する開始イベントとして機能します。接続のイベントソースの定義後は、Secure Agent 上でのみ、接続をパブリッシュできます。その後、プロセス内のイベントソースにアクセスし、イベントソースダウンストリームが生成したプロセスオブジェクトを消費する目的で、プロセスを Secure Agent 上でのみデプロイできます。

接続のイベントソースを作成するには、**[イベントソース]** タブから、**[イベントソースの追加]** をクリックします。使用可能なリストから、イベントソースタイプを選択します。

以下の表に、すべてのイベントソースタイプに使用できるイベントソースプロパティを示します。

プロパティ	説明
名前	必須。この名前は、この接続に一意である必要があります。
説明	オプション。
有効	このイベントソースをパブリッシュ後即時に使用できるようにするには、**[はい]** を選択します。 このイベントソースを使用準備ができるまで無効にするには、**[いいえ]** を選択します。

Amazon S3 イベントソースのプロパティ

Amazon S3 接続を定義してプロセスにデプロイしたら、バケットで新規オブジェクトを監視できます。S3 コネクタは、新規オブジェクトを検出すると、S3 オブジェクトを読み取り、開始イベントとして機能するイベントソースのサブスクライブ元のプロセスに、このオブジェクトを送信します。

各接続に複数のイベントソースを追加できます。

イベントをコンシュームしたりサブスクライブしたりするように定義されたプロセスがまだない場合でも、接続は、パブリッシュの直後にバケット内のオブジェクト処理を開始します。初期遅延の設定を使用すると、イベントソースからイベントを処理するためのプロセスをパブリッシュするまで、最初のバケットの処理を延期できます。

次の表に、すべての S3 イベントソースタイプに該当するファイルの場所の設定について説明します。

プロパティ	説明
S3 バケット名	必須。イベントソースで処理するオブジェクトが格納された Amazon S3 バケット名。 疑問符（?）は、このフィールドでは使用できない文字です。
S3 プレフィックス	処理するオブジェクトを決定する S3 プレフィックス。イベントソースは、この属性を使用して、共通のプレフィックスを持つ S3 オブジェクトのサブセットのみをコンシュームできます。プレフィックスが指定されていない場合、イベントソースによりバケット内のすべてのオブジェクトが読み取られます。

次の表に、すべての S3 イベントソースタイプに該当する**［オブジェクト操作の設定］**について説明します。

プロパティ	説明
S3 オブジェクトの削除	必須。このプロパティを使用して、処理済みの S3 オブジェクトをソースバケットから 1 つも削除しないか、一部を削除するか、すべてを削除するかを設定します。 次のいずれかのオプションを選択します。 - ［いいえ］: 処理済みの S3 オブジェクトを削除しません。 - ［コンシューム済み］: 正常に処理された S3 オブジェクトをすべて削除します。 - ［すべて］: 失敗したオブジェクトや正常に処理されたオブジェクトを含む、すべての S3 オブジェクトを削除します。 **注**: **［いいえ］**または**［コンシューム済み］**を選択した場合、コネクタは移動の設定を使用して、オブジェクトが 2 回コンシュームされることを防ぎます。**［移動先バケット］**に名前を設定していない場合、コネクタはデフォルトバケットを使用します。 例えば、**［いいえ］**を選択した場合、コネクタは**［移動先バケット］**と**［移動プレフィックス］**のプロパティを使用して、処理済みの S3 オブジェクトを移動します。 デフォルト: いいえ。
移動先バケット	オプション。正常に処理された S3 オブジェクトの移動先バケットを指定します。 **［移動プレフィックス］**を指定しない場合、移動先バケットにはソースバケットとは別のバケットを指定することをお勧めします。 デフォルト: ソースバケット。
移動プレフィックス	オプション。正常に処理された S3 オブジェクトのキーに追加するプレフィックスを指定します。 処理済みのオブジェクトをソースバケットに移動する場合は、移動プレフィックスを指定することをお勧めします。 デフォルト: `.ae-done/`。

プロパティ	説明
失敗の場合の移動先バケット	オプション。失敗した S3 オブジェクトの移動先バケットを指定します。 **[失敗の場合の移動プレフィックス]** を指定しない場合、失敗の場合の移動先バケットには、ソースバケットと移動先バケットとは別のバケットを指定することをお勧めします。 デフォルト: ソースバケット。
失敗の場合の移動プレフィックス	オプション。失敗した S3 オブジェクトのキーに追加するプレフィックスを指定します。 失敗したオブジェクトをソースバケットに移動する場合は、移動プレフィックスを指定することをお勧めします。 デフォルト: .ae-error/。
バックアップフォルダ	オプション。正常に処理された S3 オブジェクトのコピーを格納するローカルディレクトリを指定します。
失敗の場合のバックアップフォルダ	オプション。失敗した S3 オブジェクトのコピーを格納するローカルディレクトリを指定します。

次の表に、すべての S3 イベントソースタイプに該当する **[エラーの取扱いの設定]** について説明します。

プロパティ	説明
最大再試行間隔	フォールトが発生した場合に、イベントソースがリカバリの試行を行うまで待機する最大時間（秒単位）を指定します。 再試行の遅延間隔は、コネクタが最大再試行間隔に達するまで、指数関数的に増えます。 デフォルトの最大値: 3600 秒（1 時間）。 許可される最小値: 1 秒。 許可される最大値: 86400 秒（24 時間）。

次の表に、すべての S3 イベントソースタイプに該当するポーリングやその他の設定について説明します。

プロパティ	説明
初期遅延	S3 バケットをポーリングするまでの秒数。 デフォルト: 1 秒。 必要な初期遅延を設定します。例えば、関連するプロセスをパブリッシュする、または必要なその他の手順を実行した後で、ポーリングが開始されるようにします。
遅延	S3 バケットを次にポーリングするまでの秒数。 デフォルト: 1 秒。 S3 API 呼び出しを減らすには、ポーリングの間の遅延を長く設定します。
ポーリングあたりの最大メッセージ数	ポーリングごとに取得するオブジェクトの最大数。 デフォルト: 10。 各ポーリングで、監視が処理対象のオブジェクト数を多数検出することが懸念される場合は、負荷を分散して、パフォーマンスを管理できる値を設定します。

モニタイベントソース用のファイルコンテンツのプロパティ

モニタイベントタイプを使用して、S3 バケットで新規オブジェクトを監視し、新規オブジェクトが検出されたら通知されるようにできます。

モニタイベントソースには、イベントソースを設定し、受け取る必要がある S3 オブジェクトデータとデータ形式に基づいて異なるコンテンツタイプを処理するために使用できるプロパティが含まれます。

次の表に、すべての S3 モニタイベントソースタイプに該当する**［コンテンツタイプ設定］**について説明します。

プロパティ	**説明**
コンテンツ形式	必須。処理するコンテンツの形式を選択します。 - **［無視］**：S3 オブジェクトから直接取得できるメタデータのみが必要（コンテンツは不要）な場合に、このオプションを使用します。 - **［プレーンテキスト］**：S3 コンテンツをプレーンテキストとして取得する場合に、このオプションを使用します。 - **［バイナリ］**：テキストとその他のファイル形式の両方について、S3 コンテンツを Base64 でエンコードされた文字列に変換する場合に、このオプションを使用します。 - **［XML および JSON］**：S3 オブジェクトのコンテンツを解析し、任意のタイプのオブジェクトまたはプロセスオブジェクトのリストに変換して、それらをイベントと一緒にプロセスに送信する必要がある場合に、このオプションを使用します。 - **［添付］**：S3 オブジェクトのコンテンツをイベントの一部として取得する必要がない場合（より効率的な場合がある）に、このオプションを使用します。S3 オブジェクトのコンテンツは、生成されたイベントに添付ファイルとして含められます。
コンテンツの簡素化	必須。処理対象の XML または JSON コンテンツが有効なプロセスオブジェクト構造になっておらず、イベントソースで、プロセスオブジェクトの形式を使用するようにコンテンツの構造を変更する場合は、**［はい］**を選択します。 例えば、XML オブジェクトに、通常はスキップされる XML 属性が含まれている場合にこのオプションを有効にすると、この属性はネストされた XML 要素に変換され、有効なオブジェクトフィールドとして処理されます。
単一オブジェクトモード	必須。すべての XML コンテンツを単一のプロセスオブジェクトに変換する場合は、**［はい］**を選択します。このオプションは、XML コンテンツを扱う必要がある場合にのみ使用します。

区切りおよび固定幅コンテンツ

Amazon S3 コネクタを使用すると、1 つの区切りコンテンツまたは固定幅イベントソースを持つ新しい S3 オブジェクトを解析できます。結果をイベントとしてプロセスに提供できます。

区切りコンテンツおよび固定幅コンテンツに次のプロパティを使用できます。

プロパティ	説明
区切り文字（区切りコンテンツパーサーのみ）	必須。 区切りコンテンツの区切り文字を指定します。 区切り文字としてスペースまたはタブ文字を使用する必要がある場合、エスケープ文字（「\s」、「\t」）を使用します。
テキスト修飾子（区切りコンテンツパーサーのみ）	区切りコンテンツのテキスト修飾子を指定します。修飾子としてスペースまたはタブ文字を使用する必要がある場合、エスケープ文字（「\s」、「\t」）を使用します。
最初のレコードを無視（区切りパーサーコンテンツのみ）	コネクタが、区切りファイルの最初の行をデータ行として処理するかどうかを決定します。以下のオプションから選択できます。 - **はい**: コネクタは、区切りファイルの最初の行をデータ行として処理しません。 - **いいえ**: コネクタは、区切りファイルの最初の行をデータ行として処理します。**［いいえ］**を選択した場合は、［カラム記述子］属性を使用してカスタムヘッダーを指定する必要があります。 デフォルト値は**［はい］**です。
行を分割	区切りコンテンツの各行を個別に処理するには**［はい］**を選択し、各行をプロセスオブジェクトに変換し、それぞれに個別のイベントを生成します。 区切りコンテンツのコンテンツを一度に処理し、単一のイベントを生成するには、**［いいえ］**を選択します。

カラム記述子	処理する固定幅カラムヘッダーを定義します。値は、名前と括弧で囲んだサイズをカンマ区切りのリストとして指定します。
	区切りコンテンツパーサーの例:
	`User Name, Password, Email`
	固定幅コンテンツパーサーの例:
	`User Name(40), Password(8), Email(14)`
	カスタムオブジェクトを使用する場合、これらのカラムヘッダーは、別の一連のヘッダーを指定しない限り、プロセスオブジェクトのヘッダー名も決定します（以下を参照）。

カラムの不整合を無視	区切りコンテンツパーサーの場合、次を決定します。 - データ行の余分なカラムを無視するかどうか。 - カラムが欠落している場合に、空の文字列値があるカラムを追加するかどうか。 有効にしない場合、イベントソースは、余分なカラムまたは欠落したカラムを見つけると例外をスローします。 固定幅コンテンツパーサーの場合、次を決定します。 - 行の余分な文字を無視するかどうか。 - 定義済みのカラムに欠落した値がある場合に、空のスペースを使用して文字を追加するかどうか。 デフォルト: いいえ。

プロパティ	説明
組み込みプロセスオブジェクトの使用	組み込みプロセスオブジェクトを S3 オブジェクトレコードのプロセスオブジェクトへのマッピングに使用するには、**[はい]** を選択します。そうすると、イベントソースはプロセスオブジェクトのセットとして S3 オブジェクトコンテンツを表します。 これは、異なるコンテンツを持ち、異なるフィールドセットを使用する S3 オブジェクトを操作する場合に便利です。 区切りコンテンツの表現を簡略化し、フィールド名のリストをカスタムオブジェクトフィールドで指定する場合、**[いいえ]** を選択します。これは、各区切りコンテンツにフィールドセット（ヘッダー）があることが事前にわかる場合に適しています。イベントソースは、区切りコンテンツファイルの各レコードについて単純なプロセスオブジェクトを生成します。
カスタムオブジェクト	ファイルコンテンツを表すために使用するプロセスオブジェクトフィールド名（ヘッダー）のカンマ区切りリスト。名前は区切りファイルのヘッダーに一致する必要があります。組み込みプロセスオブジェクトを［いいえ］に設定するときは、ここに値を指定します。「index」は行インデックス情報として予約されているため、フィールド名として使用しないでください。 ファイルコンテンツを表すために使用するプロセスオブジェクトフィールド名（ヘッダー）のカンマ区切りリストを入力します。名前は区切りファイルのヘッダーに一致する必要があります。ファイルの解析時にここで入力したヘッダーが存在しない場合、関連フィールドは生成されたオブジェクト内で空になります。 組み込みプロセスオブジェクトの使用に［いいえ］を選択した場合は必須です。 **注**: ここで「index」をフィールド名として使用することはできません。これは行インデックス情報用に予約されています。 生成されたカスタムオブジェクトは提供されたフィールド名セットを使用しますが、名前は *NCName* の形式で表されます。これにより、区切りコンテンツヘッダー名に禁止文字が含まれる場合は削除し、有効なプロセスオブジェクトフィールド名であることが保証されます。

固定幅ファイル

固定幅パーサーでは、カラムヘッダーがない、または区切り文字がない固定幅ファイルを処理できます。イベントソースプロパティで、固定幅コンテンツを表すプロセスオブジェクトを生成できるカラム記述子を定義できます。

例えば、カラム記述子プロパティ内で、データファイルの構造を定義し、カラム名やカラム長のリスト（括弧で囲む）を入力できます。

```
ID(3), User Name(100), Password(20), Nick Name(25), Email(20)
```

これにより、固定幅パーサーが固定幅ファイルを読み取り、これらの値に基づいてレコードを構成するプロセスオブジェクトを生成できます。これは、カラム幅があることを加えて、区切りコンテンツパーサーと類似しています。これらの値によって、カスタムのプロセスオブジェクトを使用する場合のフィールド名セットのデフォルトも定義されます。

組み込みおよびカスタムのプロセスオブジェクトの両方が、イベントソースタイプに使用できます。

コネクタがデータの整合性を検証するため、指定したカラム記述子を超える、またはそれに足りない行が固定幅ファイル内の行にある場合、例外エラーが発生することがあります。行の余分な文字を無視し、文字数が指定に足りない行の末尾に空白を追加するには、［カラムの不整合を無視］プロパティを有効にできます。

第 4 章

Amazon S3 イベントターゲット

この章では、以下の項目について説明します。

基本的なイベントターゲットのプロパティ

定義する各接続について、イベントターゲットがプロセスから呼び出されるとき、またはファイルやメッセージの書き込みのための操作を指定する、1 つ以上のイベントターゲットを含めることができます。例えば、プロセスオブジェクトから読み取るイベントターゲットを定義し、カンマ区切りファイルに書き込めます。

接続のイベントターゲットのプロパティを設定するには、**[イベントソース]** タブから、**[イベントターゲットの追加]** をクリックします。使用可能なリストから、イベントターゲットタイプを選択します。

以下の表に、基本的なプロパティを示します。

プロパティ	説明
名前	必須。Process Designer に表示されるイベントターゲットの名前。この名前は、この接続に一意である必要があります。
説明	オプション。Process Designer で参照のために表示されるイベントターゲットの説明。

Amazon S3 イベントターゲットのプロパティ

イベントターゲットのプロパティを使用すると、オブジェクト作成を制御し、Process Designer から呼び出せるイベントサービスを作成できます。

それぞれのイベントターゲットについて、タイプ（ライタまたは区切りコンテンツライタ）を選択し、一意の名前を指定します。すべてのイベントターゲットで、次のプロパティも必要です。

Amazon S3 設定	説明
S3 バケット名	必須。Amazon S3 バケット名。ここに入力したバケット名が存在しない場合、接続プロパティに指定された Amazon S3 設定を使用し、イベントターゲットによって自動的に作成されます。 疑問符（?）は、このフィールドでは使用できない文字です。
ストレージクラス	必須。PUT オブジェクト要求に設定する S3 ストレージクラス。STANDARD または REDUCED_REDUNDANCY を選択します。詳細については、Amazon S3 ドキュメントを参照してください。

ライタプロパティ

S3 ライタを使用してイベントターゲットを定義する場合は、S3 ストレージバケットに S3 オブジェクトとしてコンテンツを書き込むことができます。追加のプロパティは必要ありません。

区切りコンテンツライタのプロパティ

S3 区切りコンテンツライタを使用してイベントターゲットを定義する場合は、カスタムプロセスオブジェクトまたは *S3DelimitedContent* オブジェクトを区切りファイル形式にシリアル化し、その結果を S3 オブジェクトとしてバケット内に保存できます。

次のプロパティは、生成された出力に対して使用できます。

区切りコンテンツの生成の設定	説明
区切り文字:	必須。区切りファイルの区切り文字。区切り文字としてスペースまたはタブ文字を指定するには、エスケープ文字（\s または\t）を使用します。
テキスト修飾子:	必須。区切りファイルのテキスト修飾子。テキスト修飾子としてスペースまたはタブ文字を指定するには、エスケープ文字（\s または\t）を使用します。
ヘッダーのスキップ	ヘッダー名なしで区切りコンテンツを保存する場合は **[はい]** を選択します。
行の末尾のスタイル	行の末尾のスタイルを選択します（Windows=\r\n、Unix=\n）。

第 5 章

Amazon S3 プロセスオブジェクト

この章では、以下の項目について説明します。

モニタイベントオブジェクト

S3 モニタイベントソースを使用する場合、新規オブジェクトがバケットに追加されるたびに、接続は、次の例のような *S3MonitorEvent* イベントオブジェクトを生成します。

注: すでに説明したように、S3 オブジェクトはバケットから削除されるため、イベントソースのプロパティでローカルフォルダを指定していない限り、ローカルファイルの情報は利用できなくなります。

イベントには、使用可能なすべての S3 オブジェクトのメタデータが含まれます。メタデータが使用できない場合、フィールドは空になります。

```
<S3MonitorEvent>
  <!-- local file information, may be empty if local backup copying is not used -->
  <localFileInfo>
    <lastModified> 2015-06-23T13:04:46.281Z </lastModified>
    <dir> D:/s3_test/s3 monitor </dir>
    <name> test1 </name>
    <path> D:/s3test/s3 monitor/test1.txt </path>
    <fullName> test1.txt </fullName>
    <ext> txt </ext>
    <size> 9 </size>
  </localFileInfo>
  <!--S3 object information -->
  <s3ObjectInfo>
    <lastModified> 2015-06-22T14:27:00Z </lastModified>
    <contentControl/>
    <s3VersionId/>
    <s3Key> test1.txt </s3Key>
    <contentType> text/plain </contentType>
    <contentEncoding> UTF-8 </contentEncoding>
    <contentDisposition/>
    <contentLength> 9 </contentLength>
    <bucketName> dvilaverde </bucketName>
    <s3ETag> b3d5cf638ed2f6a94d6b3c628f946196 </s3ETag>
  </s3ObjectInfo>
  <!-- if Content Format is equal Ignore content of the object will be skipped -->
  <!-- or if Content Format is equal PlainText content will be provided as a string field -->
    <content>S3 object content string</content>
  <!-- or if Content Format is equal Binary content will be provided as a Base64 encoded string field -->
    <content>dGVzdCBzdHJpbmc=</content>
```

```
  <!-- or if Content Format is equal XML or JSON content will be provided as a list of one or more process
objects -->
    <content>
    <lastName>Smith</lastName>
    <phone>111122-222</phone>
    <firstName>Bob</firstName>
    </content>
    <content>
    <lastName>Smith2</lastName>
    <phone>111122-222</phone>
    <firstName>Bob2</firstName>
    </content>
  <!-- or if Content Format is equal Attachment content will be provided as an attachment field -->
    <attachment>cid:29cde59c-e85b-41e0-834c-22bbed5a5b9a</attachment>
</S3MonitorEvent>
```

単一オブジェクトモードの使用

XML ドキュメントを処理するときに、ユーザーオブジェクトのリストを作成するか、1 つのプロセスオブジェクトとするかを選択できます。

例えば、次の XML には、user オブジェクトのリストが含まれます。ここで、users 要素はリストのルート要素になります。

```
<users>
  <user>
    <firstName>Bob</firstName>
    <lastName>Smith</lastName>
    <phone>111122-222</phone>
  </user>
  <user>
  <user>
    <firstName>Bob2</firstName>
    <lastName>Smith2</lastName>
    <phone>111122-222</phone>
  </user>
</users>
```

このケースで、XML ドキュメント全体を、オブジェクトリストフィールドが *user* の 1 つの *users* プロセスオブジェクトに変換する場合は、**[単一オブジェクトモード]** を有効にします。XML を *user* オブジェクトのリストに変換する場合は、**[単一オブジェクトモード]** を無効にします。

区切りコンテンツパーサープロセスオブジェクト

イベントソースが設定およびパブリッシュされたら、新しいオブジェクトについて指定した S3 バケットの監視を開始します。新しいオブジェクトが見つかったときに、コンテンツは解析され、S3 オブジェクト情報、ローカルコピー情報（ある場合）、解析されたコンテンツを表すプロセスオブジェクトのセットが含まれるイベントが生成されます。生成されたイベントは、これらのイベントをリスンするプロセスに送信されます。

前述のとおり、生成されたオブジェクトは、組み込みのプロセスオブジェクトまたはカスタムのプロセスオブジェクトとして処理できます。

これらの要因は、手段を選定するためのガイドとして使用します。

- 組み込みのプロセスオブジェクトは、異なる構造を持ち、事前にコンテンツヘッダーの内容がわからない S3 オブジェクトを処理するために使用します。
- カスタムのオブジェクトフィールドは、事前にコンテンツヘッダーの内容がわかっているときに、同じ構造を持つ S3 オブジェクトを処理するために使用します。この方法により、必要なデータのみを抽出できます。生成されたオブジェクトの方が、より簡単に扱え、プロセスで処理するコードも少なくて済みます。

どの手段を選んだ場合でも、生成されたプロセスオブジェクトの違いに注意してください。

組み込みプロセスオブジェクトの出力

各 S3 区切りコンテンツオブジェクトが見つかって処理されるときに、単一のパラメータ（*delimitedContent*）を持つイベント（*S3DelimitedContentParserEvent*）を生成し、組み込みのプロセスオブジェクト（*S3DelimitedContent*）を使用するか、カスタムのプロセスオブジェクト（*CustomS3DelimitedContent*）を使用するかを示します。

例えば、いくつかのデータを持つ単純な区切りコンテンツオブジェクトを処理する場合、ファイルが次のようになっていることがあります。

```
Country Capital Area Region
USA Washington 11111 "North America"
Ukraine Kiev 22222 Europe
Japan Tokyo 33333 "Asia Pacific"
```

S3 オブジェクトメタデータフィールドが、生成されたイベントに常に追加されます。ローカルファイルで使用できるメタデータに基づいて、生成されたプロセスオブジェクトは次のようになります。

```
<S3DelimitedContent>
  <!-- S3 object information -->
  <s3ObjectInfo>
    <lastModified>2015-06-23T14:20:22Z</lastModified>
    <contentControl/>
    <s3VersionId/>
    <s3Key>test.csv</s3Key>
    <contentType>text/plain</contentType>
    <contentEncoding>UTF-8</contentEncoding>
    <contentDisposition/>
    <contentLength>130</contentLength>
    <bucketName>myBucket</bucketName>
    <s3ETag>ddfadc6de31dd30a6588bee8c01e157e</s3ETag>
  </s3ObjectInfo>

        <!-- local copy file information, will be empty if local copying is not used -->
  <localFileInfo>
    <lastModified>2015-06-23T14:20:22.086Z</lastModified>
    <dir>D:/camel_test/s3 monitor</dir>
    <name>test</name>
    <path>D:/camel_test/s3 monitor/test.csv </path>
    <fullName>test.csv</fullName>
    <ext> csv </ext>
    <size>130</size>
  </localFileInfo>

        <!-- total number of rows in the result -->
        <totalRowsCount>3</totalRowsCount>

  <!-- list of file headers with names and indexes -->
  <header>
    <name>Country</name>
    <fieldIndex>1</fieldIndex>
  </header>
  <header>
    <name>Capital</name>
    <fieldIndex>2</fieldIndex>
  </header>
  <header>
    <name>Area</name>
    <fieldIndex>3</fieldIndex>
  </header>
  <header>
    <name>Region</name>
    <fieldIndex>4</fieldIndex>
  </header>

  <!-- delimited content records -->
  <record>
  <field>
    <value>USA</value>
```

```
  </field>
  <field>
    <value>Washington</value>
  </field>
  <field>
    <value>1111</value>
  </field>
  <field>
    <value>North America</value>
  </field>
    ...
</S3DelimitedContent>
```

生成されたイベントには、*S3DelimitedContent* プロセスオブジェクトが含まれます。ローカルコピー情報は、イベントソースに処理済みの S3 オブジェクトのローカルコピーが格納されていない場合は、空になることがあります。

生成された結果には、ソースオブジェクト、ヘッダーオブジェクトのリスト、レコードのリスト、行の合計数に関するファイル情報が格納されることがわかります。この S3 オブジェクトコンテンツは、ヘッダー名と各フィールドの値で表されます。

区切りコンテンツを行分割モードで処理する場合、S3 オブジェクトコンテンツは別々の行に分けられ、各行について、S3 接続はヘッダーと 1 つのレコードを持つ区切りコンテンツオブジェクトを生成します。

カスタムオブジェクトフィールドの出力

代わりにカスタムのオブジェクトフィールドを使用して、同じ S3 オブジェクトを処理する場合、ヘッダー名「Country, Capital, Area」（ただし「Region」はない）を提供できます。

この場合、同じオブジェクトの結果は次のようになります。

```
<AwsS3DelimitedContentParserContent>
  <!-- S3 object information -->
        <s3ObjectInfo>
    <lastModified>2015-06-23T14:20:22Z</lastModified>
    <contentControl/>
    <s3VersionId/>
    <s3Key>test2.csv</s3Key>
    <contentType>text/plain</contentType>
    <contentEncoding>UTF-8</contentEncoding>
    <contentDisposition/>
    <contentLength>130</contentLength>
    <bucketName>myBucket</bucketName>
    <s3ETag>ddfadc6de31dd30a6588bee8c01e157e</s3ETag>
  </s3ObjectInfo>

        <!-- local copy file information, will be empty if local copying is not used -->
  <localFileInfo>
    <lastModified>2015-06-23T14:20:22.086Z</lastModified>
    <dir>D:/camel_test/s3 monitor</dir>
    <name>test</name>
    <path>D:/camel_test/s3 monitor/test.csv </path>
    <fullName>test2.csv</fullName>
    <ext>csv</ext>
    <size>130</size>
  </localFileInfo>

        <!-- total number of rows in the result -->
        <totalRowsCount>3</totalRowsCount>

  <!-- custom delimited content records with specified by user fields -->
  <record>
    <index>1</index>
    <Area>11111</Area>
    <Capital>Washington</Capital>
    <Country>USA</Country>
  </record>
  <record>
    <index>2</index>
```

```
    <Area>22222</Area>
    <Capital>Kiev</Capital>
    <Country>Ukraine</Country>
  </record>
  <record>
    <index>3</index>
    <Area>33333</Area>
    <Capital>Tokyo</Capital>
    <Country>Japan</Country>
  </record>
</AwsS3DelimitedContentParserContent>
```

ここで、プロセスオブジェクトには S3 オブジェクト情報、ローカルコピー情報、レコードの合計数が格納されます。「Region」データは、カスタムオブジェクトフィールドとして指定されていないために除外されます。

カスタムオブジェクトフィールドを使用するときの注意事項:

- 生成された出力には、まだ S3 オブジェクト情報が含まれます。
- プロセスオブジェクト名は*<sourceName>Content* という形式になります。

  注: カスタムレコードオブジェクトは*<sourceName>Record* という形式になります。いくつかの区切りコンテンツイベントソースがある場合、それぞれは独自のカスタムコンテンツやレコードオブジェクトを使用します。
- フィールド名は、区切りコンテンツヘッダー名に禁止文字が含まれる場合は削除し、有効なプロセスオブジェクトフィールド名であることを保証するために、*NCName* という形式に変換されます。
- ソース S3 オブジェクトに必要なヘッダーが含まれていない場合、生成された出力内でこのフィールドは空になります。上記の例では、「Region」がカスタムオブジェクトフィールドとして指定されており、ソース S3 オブジェクトに「Region」ヘッダーが含まれていない場合、フィールドはプロセスオブジェクトに現れますが、空になります。
- 上に示すように、フィールドをスキップするには、カスタムオブジェクトフィールドのリストからこれらを除外します。

イベントターゲットを使用したコンテンツの記述

S3 用のイベントターゲット定義を含む接続をパブリッシュするときに、S3 オブジェクトを生成するためにプロセスから呼び出せるサービスを作成します。

ライタイベントターゲット

StoreS3ObjectRequest は、ここに示すように、ターゲットオブジェクトパラメータおよびコンテンツを含むプロセスオブジェクトです。これは、イベントターゲットへの入力です。

```
<StoreS3ObjectRequest>
       <!-- create S3 object metadata -->

  <s3ObjectParameters>
    <!-- object key, is required -->
    <awsS3Key>test.txt</awsS3Key>
    <!-- content type, by default text/plain type is used -->
    <contentType>text/plain</contentType>
    <!-- content control -->
    <contentControl>…</contentControl>
    <!-- content disposition which is a default file name (similar to Content-Disposition HTTP header) -->
    <contentDisposition>…</contentDisposition>
    <!-- content encoding, by default this property is empty -->
    <contentEncoding>…</contentEncoding>
    <!-- last modified timestamp in milliseconds -->
    <lastModified>…</lastModified>
    <!-- storage class (STANDARD or REDUCED_REDUNDANCY), can be used to overwrite the same attribute
```

```
        of the event target -->
    <storageClass>…</storageClass>
    <!-- canned acl (Private, PublicRead, PublicReadWrite, AuthenticatedRead, LogDeliveryWrite,
        BucketOwnerRead, BucketOwnerFullControl). -->
    <cannedAcl>…</cannedAcl>
    <!-- a well-constructed Amazon S3 Access Control List xml string. -->
    <s3Acl>…</s3Acl>
  </s3ObjectParameters>
  <!-- content format (PlainText|Binary|Attachment|XML|JSON) depending on what format you use the content
      should be provided differently -->
    <format>PlainText|Binary|Attachment|XML|JSON</format>

  <!-- this field can be used only if you set format to PlainText or Binary.
      if you use PlainText format this field should contain some textual information that will be written as
      is to the bucket; if you use Binary format here you should provide a Base64 encoded string that will be
      decoded and written to the target bucket -->
    <content>Hello World</content>

  <!-- if you chose the XML or JSON format you should use either the object or objects field to provide one
or
      several process objects that should be converted to XML/JSON content -->
    <object>some process object</object>
    <objects>a list of process objects</objects>

  <!-- optionally you can provide objectName and listName attribute values, details see below this example -->
    <objectName>user</objectName>
    <listName>users</listName>

</StoreS3ObjectRequest >
```

アクセス制御リスト（ACL）パラメータの詳細については、Amazon のマニュアルを Access Control List Overview および CannedAccessControlList で参照してください。

必須のパラメータは次のパラメータのみです。

- AwsS3Key
- format

その他すべてのパラメータは省略できます。選択する形式に応じて、パラメータは異なります。

注:

StoreS3ObjectResponse は、次のようなライタイベントターゲットが返すプロセスオブジェクトです。

```
<StoreS3ObjectResponse>

  <!-- S3 object version (if available) -->
    <awsS3VersionId/>
  <!-- S3 object ETag -->
    <awsS3ETag>b10a8db164e0754105b7a99be72e3fe5</awsS3ETag>

</StoreS3ObjectResponse>
```

XML/JSON 形式

XML/JSON 形式を使用するときは、S3 接続は *object*（単一のプロセスオブジェクト）または *objects*（プロセスオブジェクトのリスト）を予期します。objects フィールドを使用して単一のプロセスオブジェクトを渡した場合、結果は単一の項目を持つ 1 つのリストになります。object フィールドを使用して同じオブジェクトを渡した場合、結果は単一の XML/JSON オブジェクト（リストではない）になります。

省略可能なパラメータである *objectName* および *listName* を使用して、生成された XML および JSON コンテンツを以下に示すように変更できます。

XML: *user* プロセスオブジェクトのリストを XML 形式にシリアル化するときの出力は次のようになります。

```
<objects>
  <object>
    <firstName>Bob</firstName>
```

```
    <lastName>Smith</lastName>
    <phone>111122-222</phone>
  </object>
  <object>
    <firstName>Bob2</firstName>
    <lastName>Smith2</lastName>
    <phone>111122-222</phone>
  </object>
</objects>
```

ルート要素には、デフォルトで汎用名（object および objects）が付けられます。これは、Process Designer がオブジェクトリストまたは個別のオブジェクト名に関する情報を何も持っていないためです。

ルート要素名を変更する必要がある場合、*objectName* パラメータおよび *listName* パラメータを使用して要素名を指定できます。

JSON: 同じオブジェクトリストを JSON 形式にシリアル化するには、*objectName* パラメータおよび *listName* パラメータを使用して、生成された結果にラッパー JSON オブジェクトを追加します。これらのパラメータがないと、出力は有効な JSON 配列になります。

```
[{
  "firstName": "Bob",
  "lastName": "Smith",
  "phone": "1111-222-333"
},
{
  "firstName": "Bob2",
  "lastName": "Smith",
  "phone": "1111-222-333"
}]
```

JSON オブジェクトと同じプロセスオブジェクトリストは、*listName* パラメータ内に users ラッパーを追加することで、単一の配列としてシリアル化することもできます。

```
{
  “users”: [{
                "firstName": "Bob",
                "lastName": "Smith",
                "phone": "1111-222-333"
           },
           {
                "firstName": "Bob2",
                "lastName": "Smith",
                "phone": "1111-222-333"
           }]
}
```

上記のように、JSON オブジェクトに、*user* オブジェクトのリストが格納された単一のフィールドを持つ単一の JSON オブジェクトが格納されています。

単一のオブジェクトを同じ方法でラップするには、代わりに *objectName* パラメータを使用します。

添付ファイル形式

添付ファイル形式を使用する場合、方法が異なります。コンテンツを *StoreS3ObjectRequest* オブジェクトの一部として渡す代わりに、独立した *S3Writer* サービス呼び出しパラメータである *attachment* を使用できます。

区切りコンテンツライタイベントターゲット

組み込みの区切りコンテンツプロセスオブジェクトとカスタムのプロセスオブジェクトのリストの両方をシリアル化できます。

区切りコンテンツイベントターゲットを含む接続をパブリッシュすると、Process Designer は、次のプロセスオブジェクトを作成します。

- StoreS3DelimitedContentRequest

- StoreS3DelimitedContentResponse

S3DelimitedContent プロセスオブジェクトとカスタムのプロセスオブジェクトのリストのどちらをシリアル化する場合も、要求および応答はほぼ同一です。

以下に例を示します。

組み込みの区切りコンテンツプロセスオブジェクトとカスタムのプロセスオブジェクトのリストの両方をシリアル化できます。

```
<StoreS3DelimitedContentRequest>
  <!-- create S3 object metadata -->
  <s3ObjectParameters>
  <!-- object key, is required -->
    <awsS3Key>test.csv</awsS3Key>
  <!-- other optional parameters (the same as S3 Writer request parameters) -->
  </s3ObjectParameters>

  <!—delimiter char; overwrites the same setting of the event target -->
    <delimiter>;</delimiter>
  <!-- text qualifier char; overwrites the same setting of the event target -->
    <textQualifier>”</textQualifier>
  <!-- whether to skip headers; overwrites the same setting of the event target -->
    <skipHeaders>false</skipHeaders>

  <!-- built-in S3DelimitedContent process object model, in this case not used -->
    <delimitedContentObject/>

  <!-- list of custom headers -->
    <header>
  <!-- header name (custom object field name) -->
    <name>name</name>
  <!-- header index (column position index) -->
    <fieldIndex>1</fieldIndex>
    </header>
    <header>
    <name>address</name>
    <fieldIndex>2</fieldIndex>
    </header>
    <header>
    <name>phone</name>
    <fieldIndex>3</fieldIndex>
    </header>

  <!-- list of custom process objects -->
  <customObjects>
    <id>1</id>
    <name>Bill</name>
    <phone>123-11-22</phone>
    <address>Lviv</address>
  </customObjects>
  <customObjects>
    <id>2</id>
    <name>Bob</name>
    <phone>222-333-222</phone>
    <address>London</address>
  </customObjects>

</StoreS3DelimitedContentRequest>
```

注: オブジェクトメタデータのパラメータは、S3 ライタで使用するものと同じです。必要に応じて、いくつかのイベントターゲット属性（区切り文字、テキスト修飾子、スキップヘッダー）を上書きできます。

カスタムのプロセスオブジェクトを使用したヘッダーの処理

Process Designer は、カスタムのプロセスオブジェクトを使用する際に、単純なオブジェクトは自動的にシリアル化しますが、複雑なフィールド（参照やオブジェクトリスト）はスキップします。生成されたファイルには、最初のプロセスオブジェクトの単純なフィールド名を使用したヘッダーの一覧が含まれています。この場合、*delimitedContentObject* フィールド（上記を参照）は空のままにする必要があります。シリアル化の最

中に、Process Designer は、最初のプロセスオブジェクトの単純なフィールド名を使用してヘッダーの一覧を生成します。この方法は、便利なときもありますが、デメリットもいくつかあります。

- 最初のオブジェクトにオプションのフィールドがない場合、これらのフィールドは、他のすべてのオブジェクトにあっても無視されます。
- プロセスオブジェクトのフィールドが特定の順序に依存していないため、結果のオブジェクトでフィールド順序が定義されません。
- 不要なフィールドをスキップすることはできません。

これらのデメリットは、要求オブジェクトに一連のカスタムヘッダーを指定することで解消できます。Process Designer はこれらのヘッダーを使用して、指定したフィールドのみを含む区切りレコードを、指定した順序で生成します。

組み込みプロセスオブジェクトを操作する場合、要求オブジェクトを定義するときに、要求内に *S3DelimitedContent* プロセスオブジェクトを含められます。次に、生成された区切りコンテンツは、このモデルオブジェクトによって記述されたヘッダーおよびレコードを照合します。

注: *S3DelimitedContent* オブジェクト内の *s3ObjectInfo* フィールドおよび *localFileInfo* フィールドは、シリアル化中無視されるため、新しいオブジェクトを自分で作成する場合はそれらを空のままにでき、区切りコンテンツプロセスオブジェクトを別途取得してそれらのフィールドが設定される場合は維持できます。

シリアル化の結果

どの方法を使用する場合でも、各ファイルの処理後には次の結果が表示されます。

- 操作のステータス。
- オプションのメッセージ文字列。
- 処理済みレコード数と S3 オブジェクトコンテンツに正常に書き込まれたレコード数を示す 2 つのカウンタ。
- 新しい S3 オブジェクトのオブジェクトメタデータ。

StoreS3DelimitedContentResponse

応答プロセスオブジェクトには次が含まれます。

```
<StoreS3DelimitedContentResponse>
  <!-- true if success, false in case of error -->
    <success>true</success>
  <!-- textual message that may contain some warnings or errors -->
    <message>…</message>
  <!-- number of processed records (number of records that are provided in request) -->
    <processedRecordsCount>3</processedRecordsCount>

  <!-- number of successfully serialized and written to S3 object content records -->
    <writtenRecordsCount>3</writtenRecordsCount>

  <!-- created S3 object information -->
  <s3ObjectInfo>
  <!-- S3 object version (if available) -->
    <awsS3VersionId/>
  <!-- S3 object ETag -->
    <awsS3ETag>ae784bc7446c154e2f802c8440a530a0</awsS3ETag>
  </s3ObjectInfo>
</StoreS3DelimitedContentResponse>
```