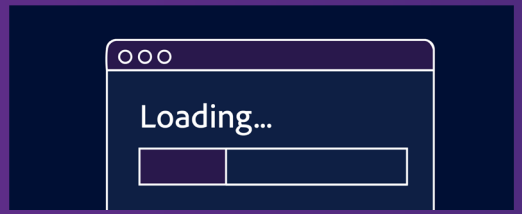
Adobe ColdFusion

2021 Release

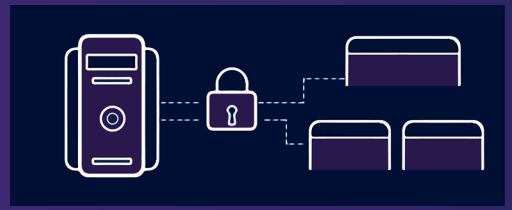
ColdFusion 2021 リリース セミナー

ColdFusion 2021 リリース トピック

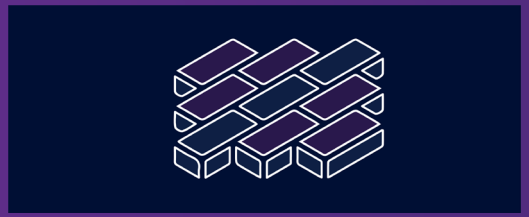


インストーラーの軽量化とモジュール化

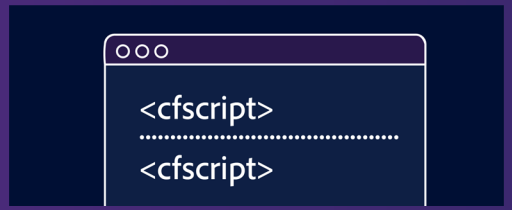
クラウドサービスの利用



SAML



パッケージマネージャー



CFML言語強化

- JavaScript経験者向けの関数追加
- データ取り扱いの機能強化
- Javaとの統合強化

インストーラーの軽量化とモジュール化

ColdFusion 2021のインストーラー

- Express インストーラー：
  - インストーラーの軽量化（150Mb）
  - 最小限のモジュールを選択しての運用
- GUI インストーラー：
  - 従来と同様の All In Oneパッケージ（1.23Gb）
  - 従来と変わらないインストール・初期（移行）ウィザード・構成（デフォルトでモジュールが全て含まれる）
  - GUIインストーラーは廃止が予定されている

Expressインストーラー

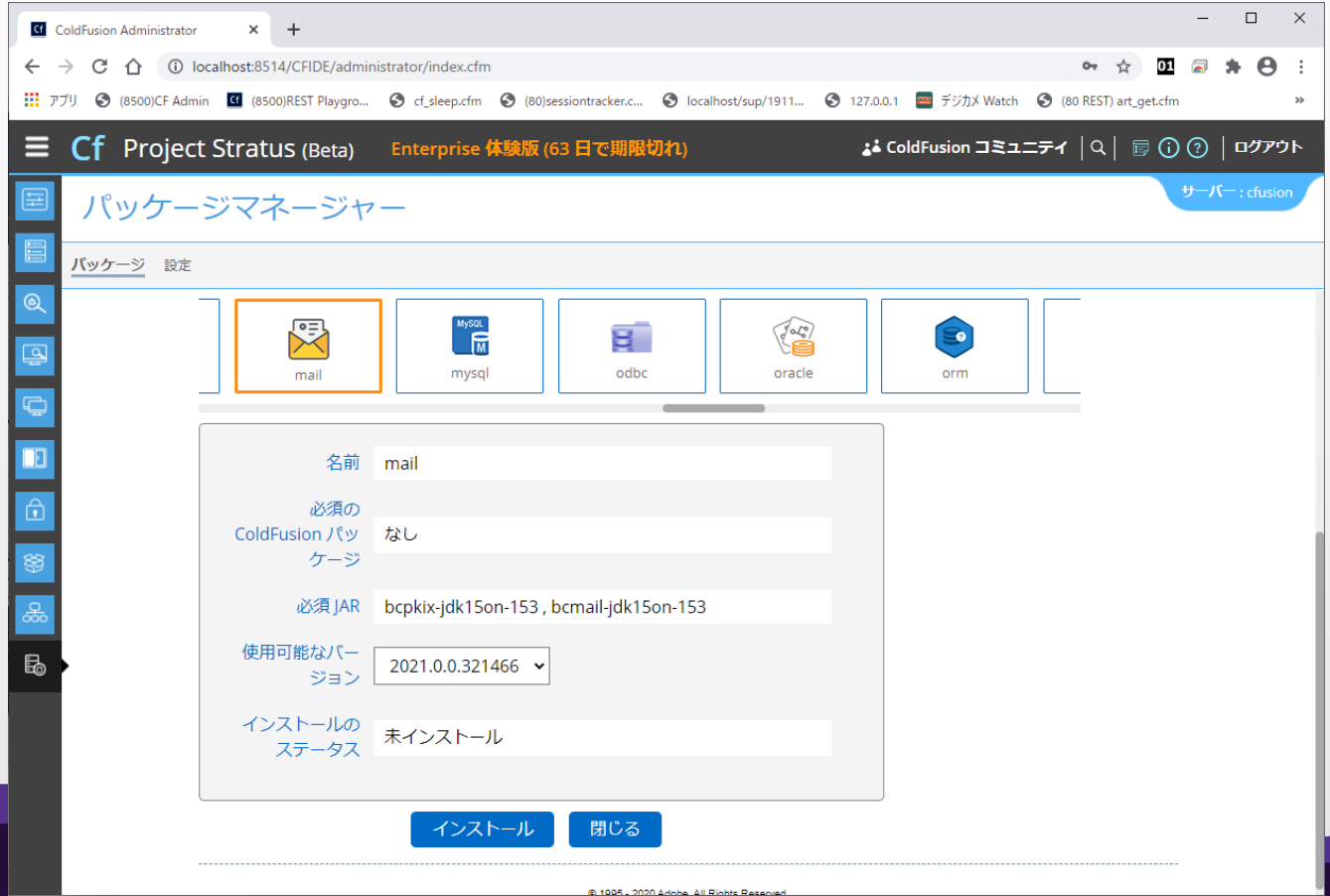
- コアコンポーネントのみ含まれる
  - インストールファイルを解凍して、フォルダを任意の場所に配置
  - cfinstall.batを実行（コンソールベースで進行）
  - 最初に最低限の設定のみを行う
  - 必要な機能（パッケージ）を自分で追加する
  - アンインストール時に注意が必要
    - （Windows）ODBCをインストール済みの場合は、ODBCをアンインストールしてから

パッケージマネージャー

パッケージマネージャーとは

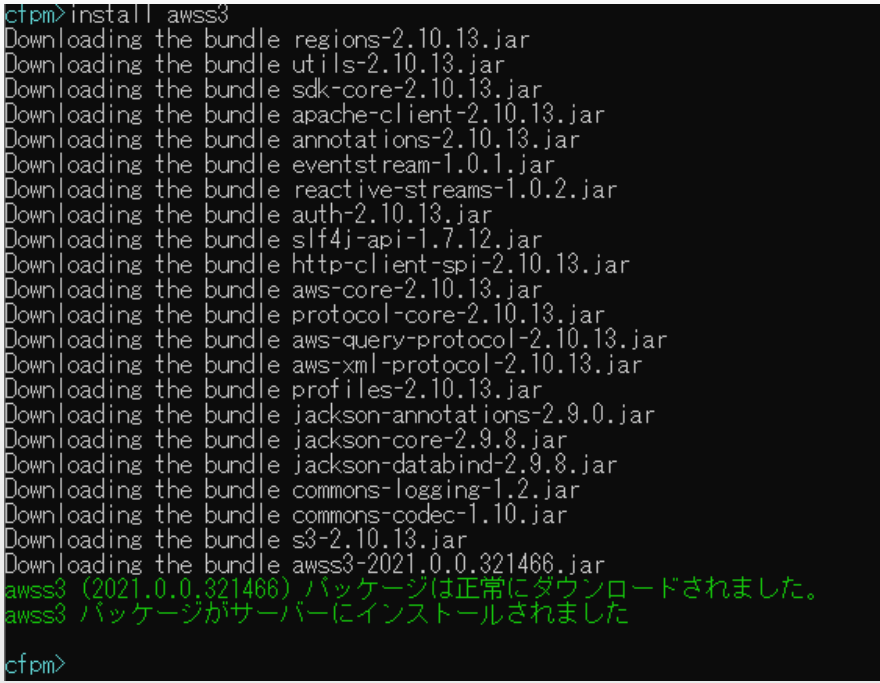
- 機能ごとに個別にパッケージを選択。リソース消費を抑える
  - コアサーバー
    - ColdFusionの基本部分。従来と同じようにUpdateが提供予定
  - インストール済みのパッケージ
    - パッケージごとにUpdateが提供される予定
    - 不要なパッケージはアンインストールも可能
  - 使用可能なパッケージ
    - 必要なパッケージはここから追加が可能

パッケージの追加方法(GUI)

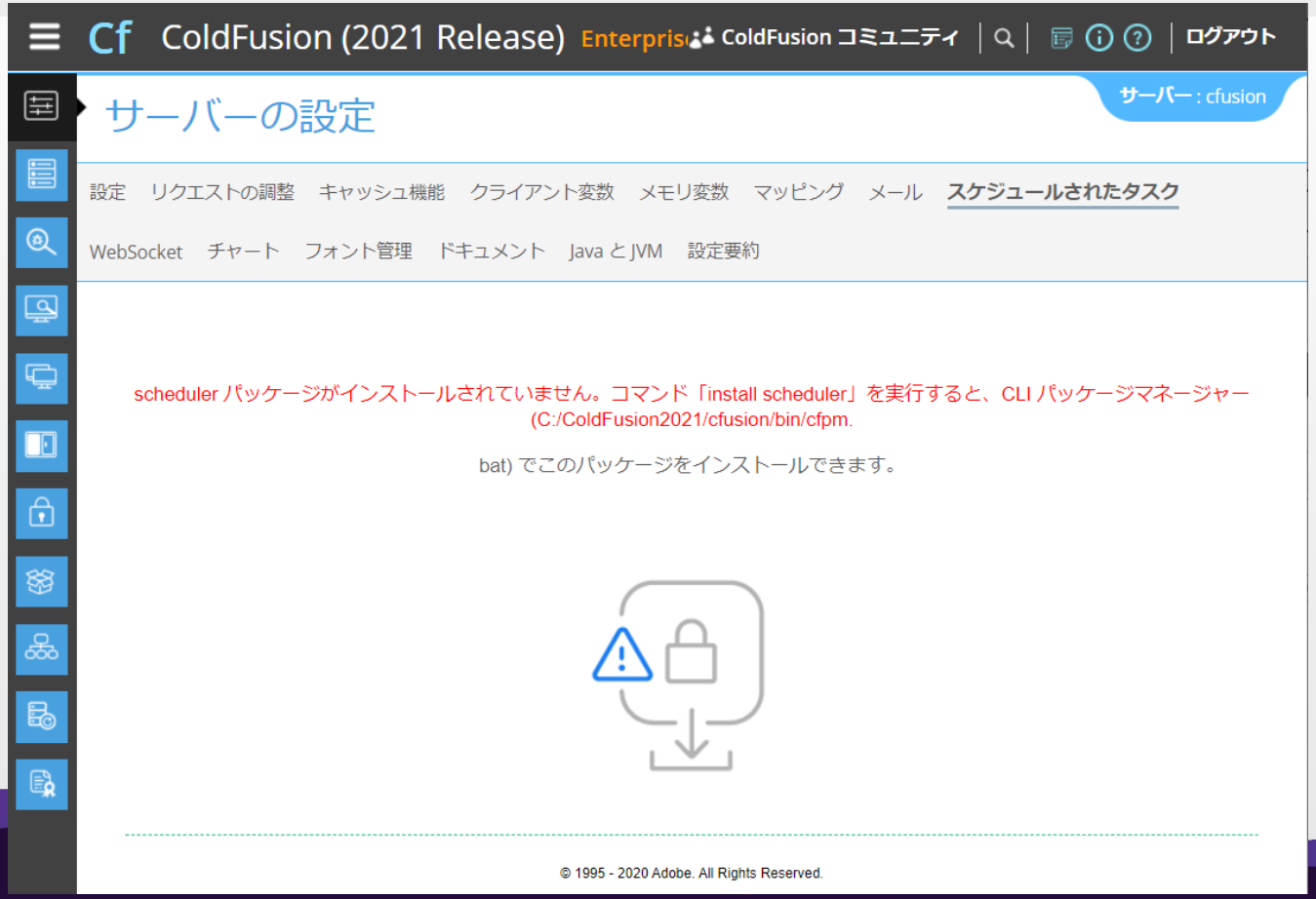


パッケージの追加方法(CUI)

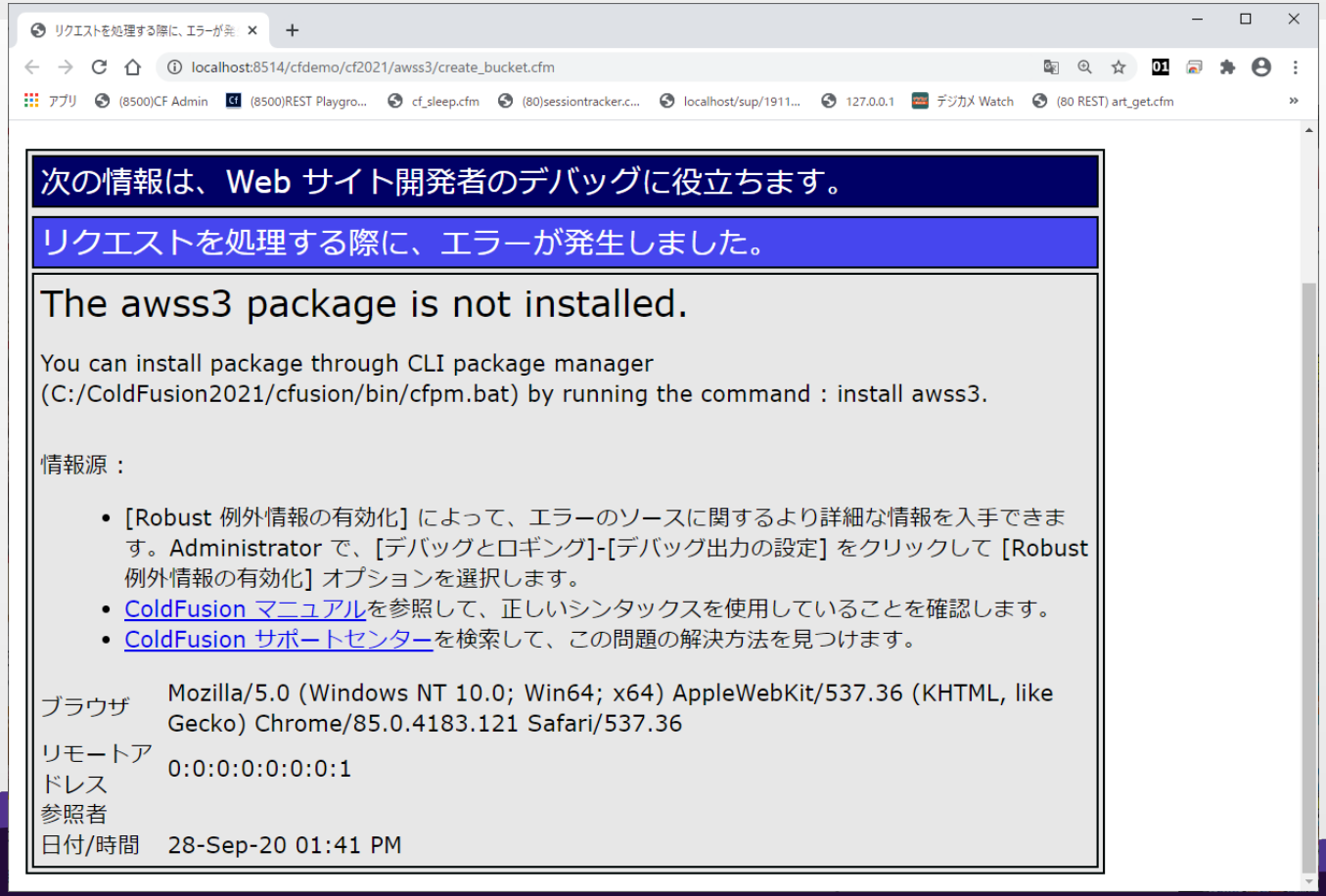
- cfpm.bat
  - list
  - listall
  - install パッケージ名 [:VERSION]
    - help 一覧を表示



パッケージが含まれていないと



プログラムのエラー例


リクエストを処理する際に、エラーが発
localhost:8514/cfdemo/cf2021/awss3/create_bucket.cfm
アプリ
(8500)CF Admin
(8500)REST Playgro...
cf_sleep.cfm
(80)sessiontracker.c...
localhost/sup/1911...
127.0.0.1
デジカメ Watch
(80 REST) art_get.cfm
次の情報は、Web サイト開発者のデバッグに役立ちます。
リクエストを処理する際に、エラーが発生しました。
The awss3 package is not installed.
You can install package through CLI package manager
(C:/ColdFusion2021/cfusion/bin/cfpm.bat) by running the command : install awss3.
情報源：
[Robust 例外情報の有効化] によって、エラーのソースに関するより詳細な情報を入手できます。Administrator で、[デバッグとロギング]-[デバッグ出力の設定] をクリックして [Robust 例外情報の有効化] オプションを選択します。
ColdFusion マニュアルを参照して、正しいシンタックスを使用していることを確認します。
ColdFusion サポートセンターを検索して、この問題の解決方法を見つけます。
ブラウザ
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36
リモートアドレス
0:0:0:0:0:0:0:1
参照者
日付/時間
28-Sep-20 01:41 PM

クラウドサービス

クラウドサービスとの連携

Microsoft AzureやAWSのストレージ、データベース、メッセージング、キャッシュ、NoSQLなどに接続

- 資格情報の設定を一元で管理
  - 複数のクラウドサービスを柔軟に呼び出しができる
    - クラウド移行時のアプリケーションの書き換え作業を軽減
- マルチクラウド機能
  - クラウドごとに異なる実行方法を、同じやり方（関数）で呼び出せるように、一部の機能に対して専用の関数を追加

※各クラウドサービスの使用料金、通信料金などは別途必要です

資格情報の設定

- 資格情報（資格情報エイリアス）
  - クラウドサービスの資格情報
- 構成（設定エイリアス）
  - それぞれのクラウドサービスの初期設定
- 資格情報 /構成の定義方法（３つ）
  - Administrator の [データとサービス]
  - Application.cfc
  - Inline

※各クラウドサービスの使用料金、通信料金などは別途必要です

クラウドサービスへのアクセス

- getCloudService()
  - service=getCloudService(資格情報,クラウド構成)
    - これらサービスへのアクセスが可能
      - Amazon SQS
      - Amazon SNS
      - Amazon DynamoDB
      - Amazon S3 storage
      - Azure Blob storage
      - Azure Service Bus

※各クラウドサービスの使用料金、通信料金などは別途必要です

ストレージサービス

Amazon S3

Azure Blob

- 主な操作
  - S3 ストレージ / Azure コンテナの作成と管理
  - ファイルのアップロードやダウンロード
  - バケット/コンテナ間のファイルコピー
  - 他、さまざまな権限等の設定

NoSQL

- これまでのRDBに加えNoSQL データベースからデータを格納・取得
  - AWS DynamoDB
  - MongoDB (Azure Cosmos DB Mongo API)

  さまざまな規模でハイパフォーマンスを提供するスキーマとフィールドを活用可能に

※各クラウドサービスの使用量、通信量などは別途掛かります

- DynamoDB

- 他のクラウドサービスと同様の設定
  - クラウド資格情報
  - クラウド構成

※各クラウドサービスの使用量、通信量などは別途掛かります

- Mongo DB

- ColdFusion Administrator [データとサービス]
  - NoSQL データソース

※各クラウドサービスの使用量、通信量などは別途掛かります

Azure Cosmos DB

- Cosmos DB の Cosmos DB API を使用したデータベース操作が可能
  - Application.cfc：アプリケーションレベルのデータソース

```
this.datasources = {
    "local" = { type="mongodb"},
    "cosmos" = {
      type="mongodb",
      host="mongodb:接続文字列",
      他、パラメーター}
}
```

メッセージ・キューサービス

- メッセージ・キューサービスを使用して、様々なコンポーネント間で非同期通信をおこなうことが可能
  - Amazon SQS、Amazon SNS
  - Azure Service Bus

※各クラウドサービスの使用量、通信量などは別途掛かります

その他サービス

- メールサービス
  - Amazon SES (SMTP)
- クラウド提供 RDSサービスの接続
  - AWS / Azure クラウドにデータベースを作成し、そこに ColdFusion 2021からの接続をサポート
    - Standard / EnterpriseがサポートするDB、バージョンが前提
- AWSサーバーレス(未リリース・2021年予定)
  - cflambda

※各クラウドサービスの使用量、通信量などは別途掛かります

CFML言語強化

(おさらい) ColdFusion 2018

他言語を経験している人にも馴染みやすく、違和感なく触っていけるように独自の記述・ルールを改良

- nullへの対応
- データ型を保持
- クエリ・配列・構造体の機能強化
- メンバ関数の強化・ (Upd5～)アロー演算子の対応
- スクリプト形式・タグ形式で同じ機能を提供
  - 過去はCFタグはタグ形式のみで使用可能、クロージャーはスクリプト形式のみ利用可能などの制約があったが改善された

JavaScript経験者向けの関数追加

標準的な関数型プログラミングを組み込み

- 厳密等価演算子、等価演算子
- スプレッド構文
- ラベルループ
- 残余（レスト）引数
- 即時実行関数（IIFE）
- 分割代入、他

厳密等価演算子（===）

- タイプ（型）チェック
  - 1 === "1" は false、1 === 1 は true
    - function returntrue(){
      return true
      }
      writeoutput(returntrue() == 'true'); // YES
      writeoutput(returntrue() == TRUE); // YES
      writeoutput(returntrue() === 'true'); // NO
      writeoutput(returntrue() === true); // YES
      writeoutput(returntrue() === TRUE); // YES

等価(==)／不等価(!=)演算子

- 等価・不等価演算子
  - ColdFusion 2018 から追加されたが、下位互換性のため等価演算子は型ではなく値で比較
  - 2021では厳密等価・厳密不等価演算子と組み合わせが可能

```
writedump(2=="2");   // YES
writedump(2==="2");   // NO
writedump('yes' == 1); // YES
writedump('yes' === 1); // NO
writedump(false ==0); // YES
writedump(false ===0);//NO
```

```
writedump(2!="2");   // NO
writedump(2!=="2");   // YES
writedump('yes' != 1); // NO
writedump('yes' !== 1);// YES
writedump(false !=0); // NO
writedump(false !==0);//YES
```

スプレッド構文

- 反復可能なオブジェクトにアクセス
- Spread演算子を使用して、反復可能なオブジェクト内のアイテムにアクセス
  - Spread演算子は ...構文 で表す
    var = [...values]

```
obj1 ={ foo: 'bar', x: 42 };
obj2 ={ foo: 'baz', y: 13 };
newObj = {...obj1,...obj2};
writeDump(newObj)
```

struct	
FOO	baz
X	42
Y	13

```
numbers = [1,2,3];
writeDump(sum( ...numbers ));

function sum(x, y, z) {
    return x + y + z;
}
```

結果：6

ラベルを用いたループ処理

- ラベルを使いbreak または continueを使ったループ文を使うことが可能
  - ネストされたループにラベルを用いる

```
for ( i = 0; i < 3; i++){
        writeoutput(i);
        for ( k = 0; k < 2; k++){
                break;
        }
}
```

結果： 012

```
LOOP : for ( i = 0; i < 3; i++){
        writeoutput(i);
        for (k = 0; k < 2; k++){
                break LOOP;
        }
}
```

結果： 0

残余（レスト）引数

- スプレッド演算子（...）と似た記述。不特定数の引数の配列を作成

```
function myFun(a,b,...otherArgs){
    writeOutput(serializeJSON(a)) → one
    writeOutput(serializeJSON(b)) → two
    writeOutput(serializeJSON(otherArgs)) → それ以降の引数
}
myFun("one","two","three","four","five","six","seven","eight")
```

出力： "one""two"["three","four","five","six","seven","eight"]

即時実行関数 (IIFE)

- 関数が作成されるとすぐに関数を実行
  - 変数宣言を分離し、グローバルスコープには影響しない

これまでの関数

```
function doSomething() {
 // ...do something...
}

doSomething = function(){
 // ...do something...
}
```

```
(function(){
    // ...do something...
})()
```

- 関数を括弧で囲むことにより、式を関数宣言ではなく関数式として評価
- 2つ目の() は、すぐに関数を実行

※IIFEでは、
式は最初の括弧のセット内で評価され
2番目の括弧のセット内で呼び出される

- 即時実行関数式(IIFE)

これまでの関数

```
// 関数の定義
  function addTogether() {
     x = 20
     y = 20
     answer = x + y
     writeOutput(answer)
  }
 // 関数の呼び出し
 addTogether()
```

IIFE

```
(function addTogether() {
      x = 20
      y = 20
      answer = x + y
      writeOutput(answer)
 })()
```

※IIFE 内は独自のスコープ
関数式で宣言された変数は
関数の外部では使用できない

分割代入(Destructuring assignment)

[変数1, 変数2, ..., 変数N] = [値1, 値2, , 値N]

val1=10
val2=20
↓
[val1,val2]=[10,20]

配列や構造体の非構造化

```
<cfset [foo, [[bar], baz]] = [61,[[42], 23]]>
<cfoutput>#bar#</cfoutput>
<cfoutput>#baz#</cfoutput>
```

出力：42 23

```
fruits=["Apple","Lemon","Orange"]
   f1= fruits[1]
   f2= fruits[2]
   f3= fruits[3]
        ↓
fruits=["Apple","Lemon","Orange"];
[f1,f2,f3] =fruits
```

```
person = {
    name: 'John Doe',
    age: 25,
    location: {
        country: 'Canada',
        city: 'Vancouver',
        coordinates: [49.2827, -123.1207]
    }
};
({name, location: {country, city, coordinates: [lat, lng]},age=40} = person)
```

構造体の非構造化では、オブジェクトプロパティを値として使用して新しい変数を作成可能

PERSON	struct	
	AGE	25
	LOCATION	struct
		CITY: Vancouver
		COORDINATES: array (1: 49.2827, 2: -123.1207)
		COUNTRY: Canada
	NAME	John Doe

struct	
AGE	25
CITY	Vancouver
COUNTRY	Canada
LAT	49.2827
LNG	-123.1207
NAME	John Doe

一括指定プロパティ(shorthand property)

- プロパティとして渡された変数と同じ名前のキーを持つオブジェクトを定義する場合は簡易表記を使用

```
<cfscript>
    CF9 = 'Centaur'
    CF10 = 'Zeus'
    CF11 = 'Splendor'
    CF2016 = 'Raijin'
    CF2018 = 'Aether'
    CF2021 = 'Project Stratus'

CFReleaseCodeNames = ${
        CF9,
        CF10,
        CF11,
        CF2016,
        CF2018,
        CF2021
    }
</cfscript>
```

casesensitive struct	
CF10	Zeus
CF11	Splendor
CF2016	Raijin
CF2018	Aether
CF2021	Project Stratus
CF9	Centaur

Javaとの統合強化

最新のオブジェクト指向プログラミングに近づける機能強化

- CFM / CFC で Java コード直接記述
- CFC：Javaインターフェースの実装/拡張
- UDF function内のJava コード

CFM / CFC で Java コード直接記述

- コード内でJavaクラスの作成/実行が可能に
  - Javaオブジェクトをインスタンス化し、CFMLブロック内にJava構造を書き込むことができる

```
classInstance = java{
    public class class1{
        public int execute (){       }
    }
}
writeoutput(classInstance.execute())
```

```
<cfjava handle="classInstance" >
    import java.io.*;
        public class Harmless{
            private String ss = “uday”;
        …}
</cfjava>
<cfset classInstance.init("content from cf")>
```

CFC：Javaインターフェース

- cfcおよびCFインターフェイスでJavaインターフェイスの機能を拡張可能に
  - 実行時に指定されたJavaインターフェイスのリストを実装するだけで、他のJavaオブジェクトと同じように動作できる
  - CFインターフェイスでJavaインターフェイスを拡張。Javaオブジェクトを予期していたメソッドにCFCを直接渡すことも可能

Component implements = “java:java.util.List, Test.AnotherCFInterface, java:com.adobe.MyInterface”
{
　　インターフェイスにリストされているすべての抽象メソッドに必須の
　　実装を提供するか、onMissingMethodを実装
}

<cfinterface extends = "java:java.util.Map">
</cfinterface>

<cfcomponent implements = "java:java.util.Map">
</cfcomponent>

UDF function内のJava コード

```
<cfset x = custName('John', 'Doe')>
<cfoutput>#x#</cfoutput>

<cffunction name="custName" type ='java'>
    <cfargument name="customerID" required="false" restargsource="Path" type="string"/>
    <cfargument name="name" required="false“ restargsource="Path" type="string"/>
    return new java.lang.StringBuffer(customerID).reverse().toString() + name;
</cffunction>
```

（スクリプト形式）

```
function custName(string customerID, string name) type=”java” {
    return new java.lang.StringBuffer(customerID).reverse().toString() + name;
}
```

その他

- イテレーター（反復子）
- CFCの静的フィールド・メソッドのサポート
- 関数の追加や強化
  - <cfquery>, QueryExecute の returnType属性
  - 配列、文字列操作系、構造体、Base64Url エンコード 他
    - 多くの機能強化や機能追加
- アプリケーションレベルのタイムゾーン

イテレーター（反復子）

- CFCでUDF をイテレーターとして、または__iter__()と配列・構造体・クエリなどを使用可能

```
arrayIteratorObj= new arrayAsIterator()

for(i in arrayIteratorObj ) {
    writedump(i)
}
```

出力：1 2 3 4 5 6

arrayAsIterator.cfc

```
component {
        public any function __iter__() {
                return [1,2,3,4,5,6];
        }
}
```

CFC：静的フィールド・メソッド

同じCFCからインスタンスを複数作成した際の共通の変数などが必要なブロック/変数/メソッドに staticを宣言

Case1.cfc

```
component {
    writeDump("non static")
    static {
        writeDump("static")
    }
}
```

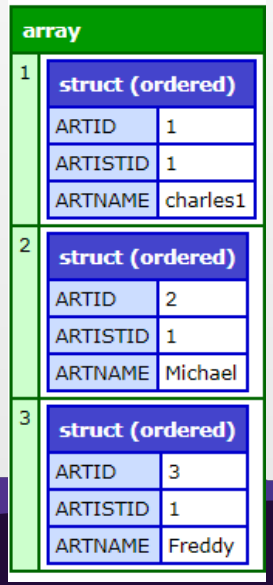
Case1.cfm

```
<cfscript>
    new Case1()
    new Case1()
</cfscript>
```

出力
static
non static
non static

cfquery / queryExecute 強化

- データベースから取得したデータを配列、JSON/配列で取得が可能
  - returntype="array | json/array | query | struct | json/struct"



array

	struct (ordered)	
1	ARTID	1
	ARTISTID	1
	ARTNAME	charles1
2	struct (ordered)	
	ARTID	2
	ARTISTID	1
	ARTNAME	Michael
3	struct (ordered)	
	ARTID	3
	ARTISTID	1
	ARTNAME	Freddy

[{"ARTID":1,"ARTISTID":1,"ARTNAME":"charles1"},
{"ARTID":2,"ARTISTID":1,"ARTNAME":"Michael"},
{"ARTID":3,"ARTISTID":1,"ARTNAME":"Freddy"}]

構造体：大文字と小文字を区別する

- mystruct = StructNew("casesensitive")
  - これまではColdFusion Administrator の「シリアル化用の構造体キーで大文字小文字を保持」設定のみ

```
animals=StructNew("casesensitive")
animals.Aardwolf="Proteles cristata"
animals.aardvark="Orycteropus afer"
animals.Alligator="Mississippiensis"
animals.albatross="Diomedeidae"
```

casesensitive struct	
Aardwolf	Proteles cristata
Albatross	Diomedeidae
aardvark	Orycteropus afer
alligator	Mississippiensis

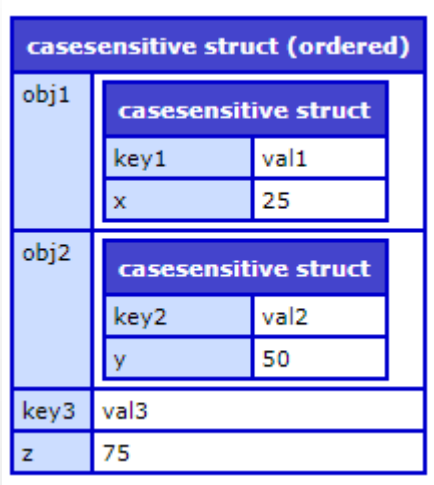
```
animals=${
        Aardwolf:"Proteles cristata",
        aardvark:"Orycteropus afer",
        alligator:"Mississippiensis",
        Albatross:"Diomedeidae"
}
```

構造体：マージ

- 構造体のマージに対応

```
obj1 =${ key1: 'val1', x: 25 };
obj2 =${ key2: 'val2', y: 50 };

mergedObj = $[obj1, obj2, "key3":"val3", z:75]
writeDump(mergedObj);
```



アプリケーションレベルのタイムゾーン

これまではOSのタイムゾーンか-Duser.timezoneでグローバルな設定しか行えなかった

- Application.cfc でアプリケーションレベルで指定が可能
  - this.timeZoneが指定可能。任意のタイムゾーンを設定可能
  - 日付/時刻関数を実行すると、タイムゾーンに沿った結果が返される

saveContent(<cfsavecontentのスクリプト式)

- スクリプト式のsaveContentの構文を見直し

```
result1 = savecontent {
        result2= savecontent {
            result3 = savecontent {
                WriteOutput("[inner1] savecontent ");
            }
            WriteOutput(result3 & "[inner2] savecontent ");
        }
        WriteOutput(result2 & "[outer] savecontent ");
}
```

RESULT1	[inner1] savecontent [inner2] savecontent [outer] savecontent
RESULT2	[inner1] savecontent [inner2] savecontent
RESULT3	[inner1] savecontent

動的 switch - case

- switch、case の値を動的に指定可能

```
s = 123
switch(((()=> 125)())) {
        case -(-s) + 2:
        writeoutput("22")
        break;
}
```

結果：22

```
<cfswitch expression="#(() => 1)()#">
   <cfcase value="5">I like kiwi!</cfcase>
   <cfcase value="#(5 GT 1)#">I like Orange!</cfcase>
   <cfdefaultcase>Fruit, what fruit?</cfdefaultcase>
</cfswitch>
```

結果：I like Orange!

SAML

SAML

SAML 2.0 をサポート。一つの資格情報で複数のアプリケーションにサインインが可能に

- サービスプロバイダー(SP) 情報
  - Administrator [セキュリティ]> [SP構成]
  - AdministratorでSPの作成が可能
- ID プロバイダー(IDP) 情報の管理
  - Administrator [セキュリティ]> [IDP構成]

InitSAMLAuthRequest

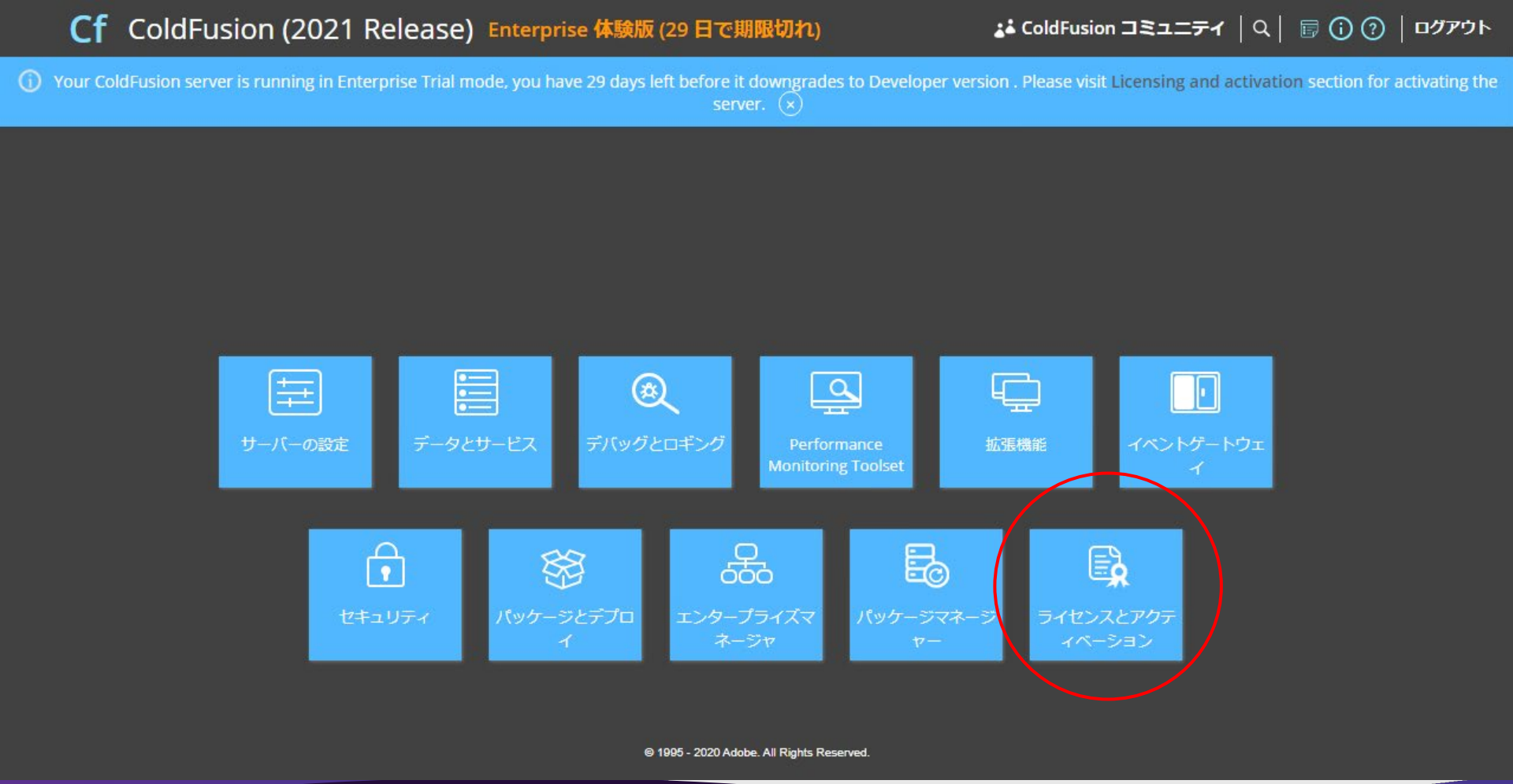
- SSO リクエスト
  - <cfset config = { idp = {name = “idp1”},
    sp = {name = “sp1”}, ... 他パラメータ}>
    <cfset InitSAMLAuthRequest(config)>
- SLO リクエスト
  - 上記＋sessionindex、nameId、nameIdFormat など、ログイン時に受け取った詳細情報を指定
    <cfset InitSAMLLogoutRequest(config)>

アクティベーション

アクティベーション

- CF2021よりアクティベーションが必要（2020.12.8更新）
  - シリアルを入力してインストール
    - オンラインでライセンスサーバーに接続できない場合などは、デベロッパーモードで起動する
    - シリアル等に問題が無く、起動時にオンライン状態の場合は自動でアクティベートが実行される
- アクティベーションに必要なのはシリアル番号のみ
- マシンを変更する際は、アンインストールする

https://helpx.adobe.com/jp/coldfusion/using/coldfusion-licensing-activation.html


Cf ColdFusion (2021 Release) Enterprise 体験版 (29 日で期限切れ)
ColdFusion コミュニティ
ログアウト
Your ColdFusion server is running in Enterprise Trial mode, you have 29 days left before it downgrades to Developer version . Please visit Licensing and activation section for activating the server.
サーバーの設定
データとサービス
デバッグとロギング
Performance Monitoring Toolset
拡張機能
イベントゲートウェイ
セキュリティ
パッケージとデプロイ
エンタープライズマネージャ
パッケージマネージャー
ライセンスとアクティベーション
© 1995 - 2020 Adobe. All Rights Reserved.

ライセンスとアクティベーション

アクティベーション　使用状況　設定

ライセンスとアクティベーションページでは、所有している ColdFusion ライセンスを管理し、インスタンスの使用状況を追跡できます。

サーバーエディション： 評価

アクティベーションステータス： 該当なし

デバイス ID： 0c3b2dd82b1bbe63f0

デプロイメントタイプ： Development

ライセンスキー

入力するシリアル番号に応じて、さまざまな機能のオン／オフが切り替わります。

新しいシリアル番号 [ ] – [ ] – [ ] – [ ] – [ ] – [ ]

アクティベート　クリア

アクティベーションで収集される情報

要素	説明
デバイス ID	ColdFusion がインストールされているホストコンピューターの ID。
新規および以前のシリアル番号	バージョンのアップグレードまたはアップデートをおこなう場合は、両方のシリアル番号が収集されます。
ColdFusion バージョン	インストールされている ColdFusion のバージョン。
デプロイメントタイプ	開発、本番、障害回復、テスト、ステージングのいずれか。
コア数	コア（物理または仮想）の数。
複数のインスタンスが作成されるかどうか	true または false
Docker デプロイメント	true または false
クラウド（例：AWS や Azure など）にデプロイされるかどうか	true または false
デプロイ先のクラウドプラットフォーム	例えば、AWS や Azure など。

デプロイメントタイプ

- インストール時にデプロイメントタイプを指定
  （現時点では、後から変更できない）
  - 本番
  - 開発
  - ステージング
  - テスト
  - 障害回復

開発、ステージング、テストなどは実行環境・ユーザーに制約あり（要EULA確認）

オンラインアクティベーション

- Administrator「ライセンスとアクティベーション」
  - ネットワーク内のファイアウォール・プロキシで特定の URL へのアクセスが許可されている必要がある
    - https://coldfusion.adobe.io
    - https://cfactivation.adobe.com/
  - プロキシは、「パッケージマネージャー／設定ページ」で指定する

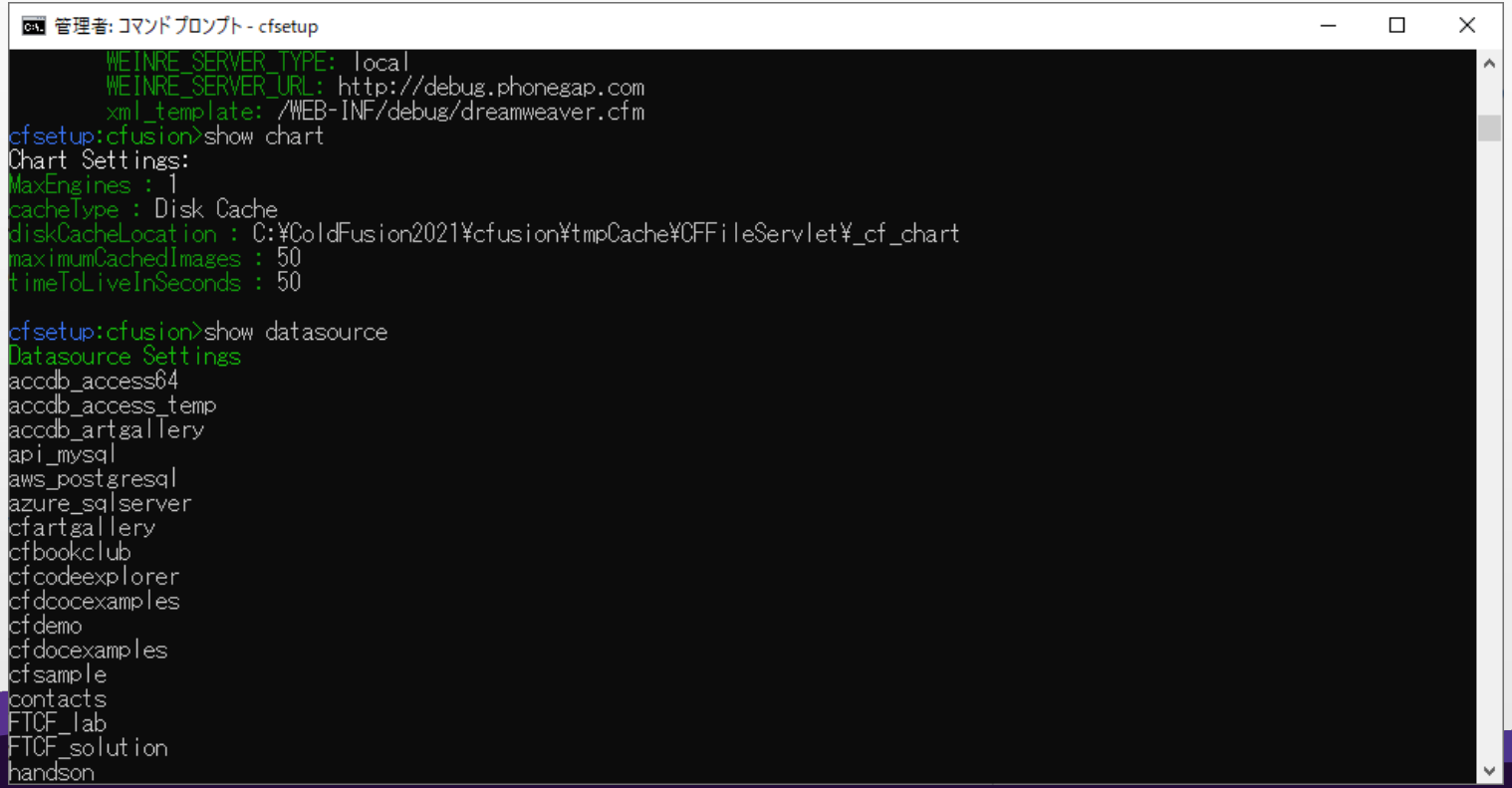
オフラインアクティベーション

1. アクティベーション用のリクエストファイル
   - 「ライセンスとアクティベーション」内、「アクティベーションリクエストを生成」
2. インターネットに接続可能なPCにて、ファイルをアップロード
   - https://www.adobe.com/go/coldfusion-activate_jp
3. 生成された応答ファイルをダウンロード
   - 「ライセンスとアクティベーション」の所定欄にアップロード
     - 72 時間以内に行う必要あり

その他

cfsetupユーティリティ

- Administratorの設定をCUIで確認・設定



廃止・非推奨の機能

https://helpx.adobe.com/jp/coldfusion/user-guide.html/coldfusion/deprecated-features.ug.html
※現時点で日本語のページは更新されていないため、英語版

- CF2021で廃止された機能：
  - CORBA, LCDS, Flash関連の機能
  - (Enterprise)cfschedule の onMisfire の fire_now

サポートプラットフォーム

https://helpx.adobe.com/pdf/coldfusion2021-support-matrix.pdf

- ColdFusion 2018と同様に64ビット版のみリリース
- 主なサポートOS
  - Windows 2012R2, 2016, 2019 Server OS
  - Red Hat Enterprise Linux 8.x, CentOS 8.x, Ubuntu 20.04
  - Mac 10.15.x
  - Solaris 11.4
- 他、データベースサポート

サポートライフサイクル、他

- ColdFusion 2018/2016 と同様の6年のサポート期間

  https://helpx.adobe.com/support/programs/eol-matrix.html

  - コアサポート5年、延長サポート1年

ColdFusion	2016		2/16/2016	2/17/2021	2/17/2022
ColdFusion	2018		7/12/2018	7/13/2023	7/13/2024
ColdFusion	2021		11/11/2020	11/10/2025	11/10/2026

- ColdFusion Builderはリリースされない
  - 2021年に Microsoft VS Codeベースの Builderを予定

ライセンスの数え方資料について

https://www.samuraiz.co.jp/coldfusion 内にPDF公開

主なポイント（2020.11現在）

- 有償・無償ライセンスの定義は従来通り
- ライセンスの数え方、価格もColdFusion 2018と同じ

- ColdFusion 2018 からの主な変更点
  - メディア提供の廃止