

ES/1 NEO

CSシリーズ

CS Scripting Interface 使用者の手引き

第22版 2022年2月

©著作権所有者 株式会社 アイ・アイ・エム 2022年

© COPYRIGHT IIM CORPORATION, 2022

**ALL RIGHT RESERVED. NO PART OF THIS PUBLICATION MAY
REPRODUCED OR TRANSMITTED IN ANY FORM BY ANY MEANS,
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY RECORDING,
OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM WITHOUT
PERMISSION IN WRITING FROM THE PUBLISHER.**

“RESTRICTED MATERIAL OF IIM “LICENSED MATERIALS – PROPERTY OF IIM

目次

第 1 章 CS Scripting Interface チュートリアル	1
1.1. 概要	1
1.2. 拡張モジュールの作成	1
1.3. 属性の追加	2
1.4. main 関数の追加	2
1.5. 拡張モジュールの実行と結果確認	3
1.6. パフォーマンスデータへのアクセス	5
1.7. db オブジェクトの属性	6
1.8. 時間の扱い	7
1.9. 欠損値の扱い	7
1.10. チューニングヒントの作成	8
1.11. パラメータの活用	9
1.12. 資源ログの作成	11
1.13. 複数システム	13
1.14. 複数システム(main_init/main_term)	16
1.15. エラーへの対処	19
1.16. 実行可能なサンプルスクリプト	20
第 2 章 CS Scripting Interface リファレンス	24
2.1. はじめに	24
2.2. 拡張モジュール	24
2.2.1. パフォーマンスデータの取り扱い	24
2.2.2. 拡張モジュールの名前	25
2.2.3. 拡張モジュールでの日本語の使用	25
2.2.4. 拡張モジュールの属性	25
2.3. db オブジェクト	33
2.3.1. RDB への直接アクセス	33
2.3.2. db オブジェクトの属性	36
2.3.3. db オブジェクトのメソッド	38
2.4. script_db オブジェクト	43
2.4.1. script_db オブジェクトの取得	43
2.4.2. script_db オブジェクトの属性	43
2.4.3. script_db オブジェクトのメソッド	43
2.5. es1lib モジュール	45
2.5.1. グループの作成	45
2.5.2. 重要度メッセージの作成	47
2.5.3. 数値データの作成	55
2.5.4. 相関度メッセージの作成	65
2.5.5. 相関判定ナビゲーションとの連携	71
2.5.6. グラフ作成	80
2.5.7. メール送信	95
2.6. CS 標準提供処理が使用するグループ名	99
2.6.1. 重要度メッセージ、相関度メッセージを分類するグループ名一覧	99

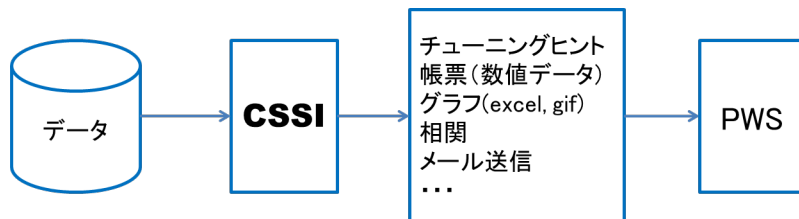
2.6.2. 数値データを分類するグループ名一覧	100
2.7. その他の es1lib モジュールの属性	104
2.7.1. es1_setenv オブジェクト	104
2.7.2. es1_timebase	107
2.7.3. es1_strftime(ts, fmt=u"%Y/%m/%d-%H:%M", timebase=None)	107
2.7.4. es1_ptl(L, pos)	108
2.7.5. es1_ptls(L, *posl)	108
2.7.6. es1_plus(values)	109
2.7.7. es1_sum(values)	109
2.7.8. es1_minus(values)	110
2.7.9. es1_multi(values)	110
2.7.10. es1_div(numer, denom)	111
2.7.11. es1_cnt(values)	111
2.7.12. es1_avg(values)	112
2.7.13. es1_min(values)	112
2.7.14. es1_max(values)	112
2.7.15. es1_ratio(numer, denom, over=100)	113
2.7.16. es1_ratio_rev(numer, denom)	113
2.7.17. es1_is_pvm_os(db)	114
2.7.18. es1_is_instance_profile()	114
2.7.19. es1_get_oracle_profile(disptext="")	114
2.7.20. es1_get_symfoware_profile(disptext="")	115
2.7.21. es1_get_sqlserver_profile(disptext="")	115
2.7.22. es1_get_db2_profile(disptext="")	115
2.7.23. es1_get_saperp_profile(disptext="")	115
2.7.24. es1_exec_error(msg)、es1_exec_cancel(msg)	116
2.7.25. es1_loginfo(msg)、es1_logwarn(msg)	116
2.8. レコード/フィールドオブジェクトの例外規定	117
2.9. esutil モジュール	118
2.9.1. イベントログの作成	118

第1章 CS Scripting Interface チュートリアル

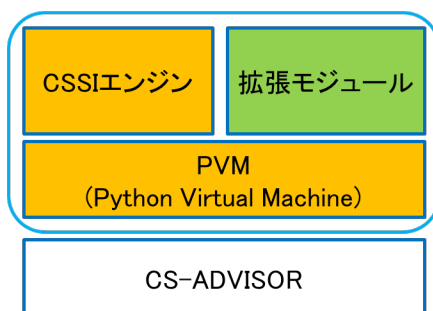
ES/1 NEO CS シリーズは、CS Scripting Interface（以降、CSSI）という評価分析機能を提供しています。CSSI を利用することで、自ら定義した独自の処理を実行することが可能になります。

1.1. 概要

CSSI の主な用途は、各種データ収集機能により蓄積されたパフォーマンスデータを入力とし、チューニングヒント／帳票／グラフなど評価分析に有用な情報を出力することです。出力物は Performance Web Service 上で閲覧することができます。



CSSIはPython2.7(プログラミング言語)を基盤としています。利用者はPythonの文法規則に従ったプログラムテキスト（以降、拡張モジュール）を作成し、独自の処理を定義します。作成した拡張モジュールは CS-ADVISOR、CSSI エンジンを通じて実行されます。CS-ADVISOR は CSSI の GUI インターフェースであり、設定や実行時に使用します。



次項から基本的な拡張モジュールの作成及び実行方法を記載します。

API や属性などの詳細は「第 2 章 CS Scripting Interface リファレンス」を参照してください。

1.2. 拡張モジュールの作成

拡張モジュールはCS インストールフォルダ以下の"_extproc"フォルダ内に作成します。ファイル名は"x_"のプレフィックスで始まり、".py"の拡張子で終わり、プレフィックスと拡張子の間は英数字のみ許可されています。日本語を使用する場合は、文字エンコード形式を utf-8 としてファイルを保存します。

C:¥IIM¥CS¥_extproc¥x_test.py

```
# -*- coding: utf-8 -*-  
from es1lib import *
```

※1 行目は文字コード宣言、2 行目は CSSI の各 API を使用するための import 文です。

1.3. 属性の追加

拡張モジュールはその動作を制御するための様々な属性が用意されています。今回は、以下とします。

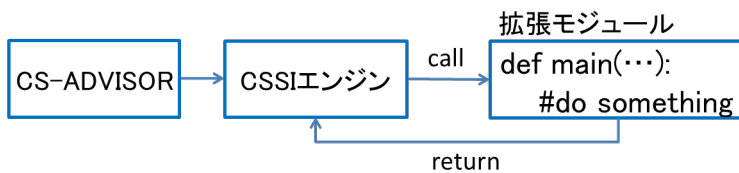
- ✓ GUI に表示される拡張モジュールの名前：test
- ✓ 対象とするシステムの数：1
- ✓ 対象とするパフォーマンスデータの種類：すべて

```
# -*- coding: utf-8 -*-
from es1lib import *

proc_desc = u'test'
proc_type = 1
proc_std = False
proc_custom = True
proc_reqsym = []
proc_loadsym = []
```

1.4. main 関数の追加

main 関数は CSSI エンジンから呼び出されるエントリーポイントです。main 関数内に処理(ロジック)を記述します。



今回は「何もしない」空の拡張モジュールです。

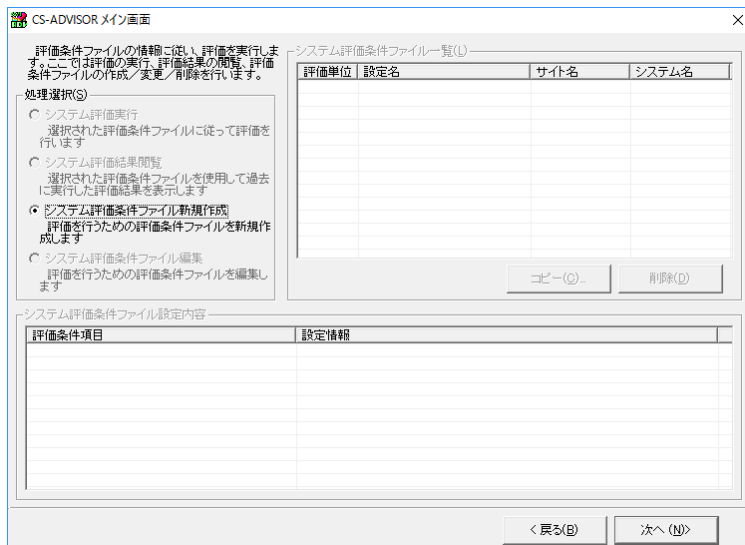
```
# -*- coding: utf-8 -*-
from es1lib import *

proc_desc = u'test'
proc_type = 1
proc_std = False
proc_custom = True
proc_reqsym = []
proc_loadsym = []

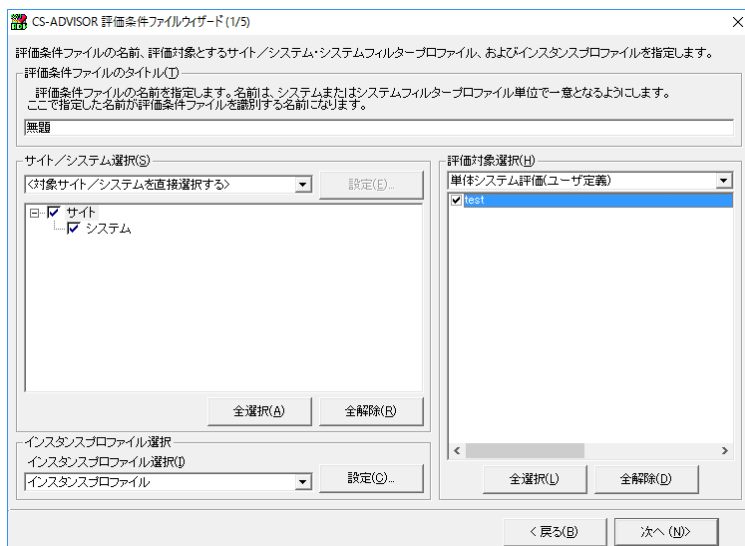
def main(context, db, param):
    pass
    # do something
```

1.5. 拡張モジュールの実行と結果確認

「ES/1 NEO CS シリーズ」から「CS」を起動し、「システム評価」を押下します。



処理選択の「システム評価条件ファイル新規作成」をチェックし、「次へ」を押下します。

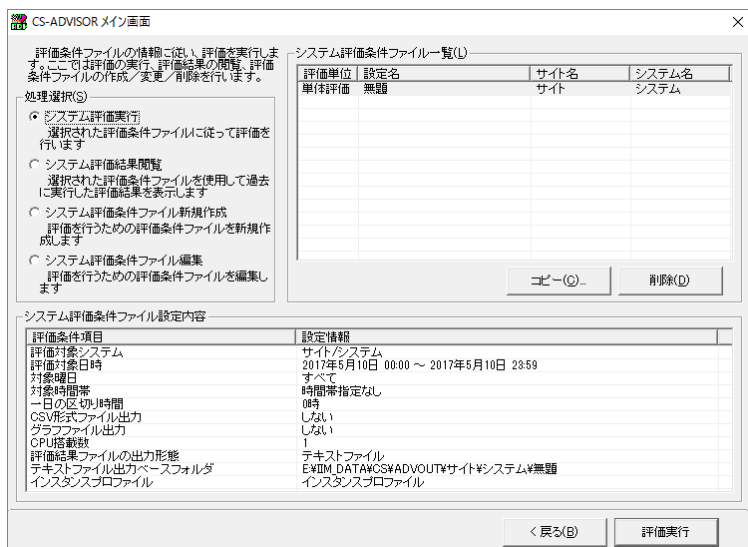


「サイト／システム選択」から対象とするシステムをチェックします。

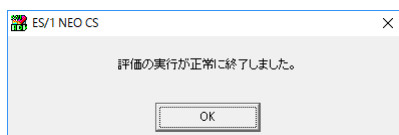
「評価対象選択」の「単体システム評価(ユーザ定義)」に作成した拡張モジュール「test」が表示されていることを確認し、チェックします。

「次へ」を押下し、対象時間を指定し、システム評価条件ファイルを登録します。

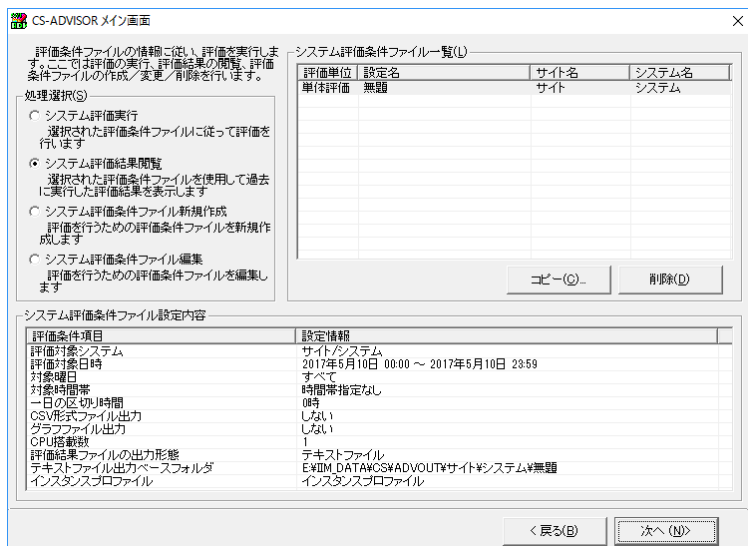
「システム評価実行」にチェックを入れ、登録したシステム評価条件ファイルを選択し、「評価実行」を押下します。
作成した拡張モジュールが実行されます。



処理が正常に終了すると以下の画面が表示されます。処理中にエラーが発生するとそのエラー内容が表示されます。



実行結果を確認する場合は「システム評価結果閲覧」にチェックを入れ、「次へ」を押下します。



「何もしない」空の拡張モジュールのため、空の評価結果が表示されます。

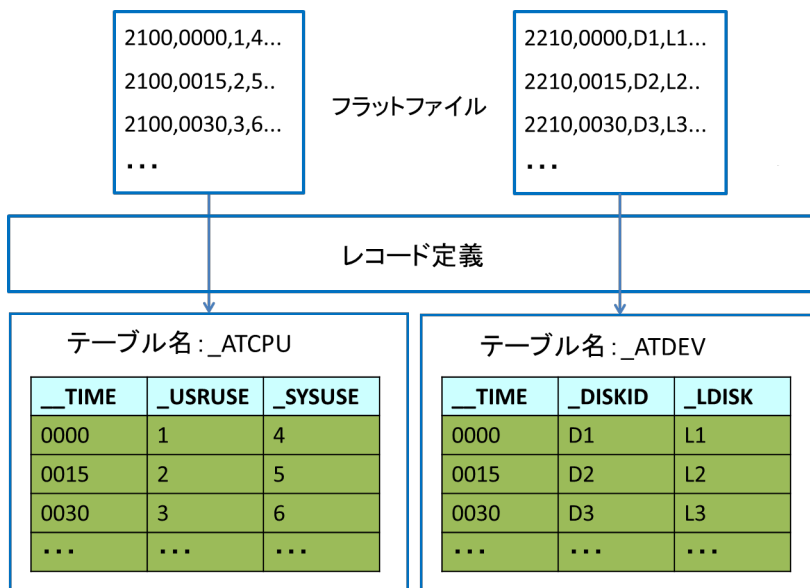


1.6. パフォーマンスデータへのアクセス

パフォーマンスデータはフラットファイルと呼ばれる CSV 形式のファイルに格納されています。処理を実行すると CSSI エンジンがフラットファイルを読み込み、内部データベース(sqlite)に展開します。



内部データベースの中にはデータの種類(レコード)毎のテーブルが存在し、各行に値が格納されています。この例ではプロセッサ(_ATCPU)とデバイス(_ATDEV)の 2 つのテーブルになります。



テーブル名、列名、値の意味については、「CS-MAGIC 使用者の手引き」第 9 章 添付資料 A. ES/1 NEO CS シリーズのクエリーで使用可能なデータ列名」を参照してください。CSSI が使用するテーブル名、列名は、先頭にアンダースコアを付与した名前になります。マニュアルの「表名: ATCPU」は「テーブル名: _ATCPU」、「列名: USRUSE」は「列名: _USRUSE」となります。

第9章 添付資料 A. ES/1 NEO CS シリーズのクエリーで使用可能なデータ列名

ES/1 NEO CS シリーズのクエリーで使用可能なデータ名(表名・列名)は以下の通りです(表中 Unix で○になっているデータでも OS の種類によっては出力されない場合があります)。

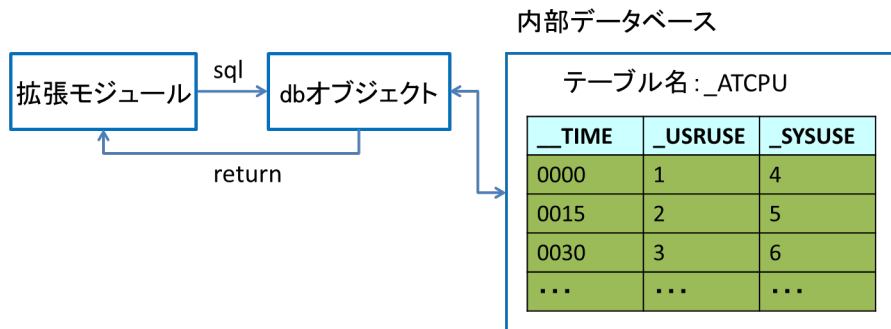
型 F: 浮動小数点型
I: 整数型
S: 文字型

9.1. システムデータ

9.1.1. プロセッサ (表名:ATCPU)

列名	Unix	Windows	型	説明
USRUSE	○	○	F	ユーザモード使用率
SYSUSE	○	○	F	カーネルモード使用率
IOWAIT	○		F	I/O Wait 率

内部データベースは db オブジェクトを介して拡張モジュールからアクセスします。具体的には db オブジェクトに SQL を渡すことで各テーブルの値を取得します。



今回は、以下の値を取得します。

- ✓ テーブル名: _ATCPU (プロセッサ)
- ✓ 列名: _USRUSE (ユーザモード使用率)
- ✓ 列名: _SYSUSE (カーネルモード使用率)

db オブジェクトは main 関数の第 2 引数です。

db オブジェクトの getcursor 関数に SQL 文を渡し、for を用いて一行ずつ値を取得しています。

```
def main(context, db, param):  
    sql = "select _USRUSE, _SYSUSE from _ATCPU"  
    for row in db.getcursor(sql):  
        usruse = row[0]  
        sysuse = row[1]
```

データベースから取得した値の型(type)は float、int、unicode のいずれかになります。どの型になるかは前述のマニュアルに記載されています。

- ✓ マニュアル表記: F = float
- ✓ マニュアル表記: I = int
- ✓ マニュアル表記: S = unicode

1.7. db オブジェクトの属性

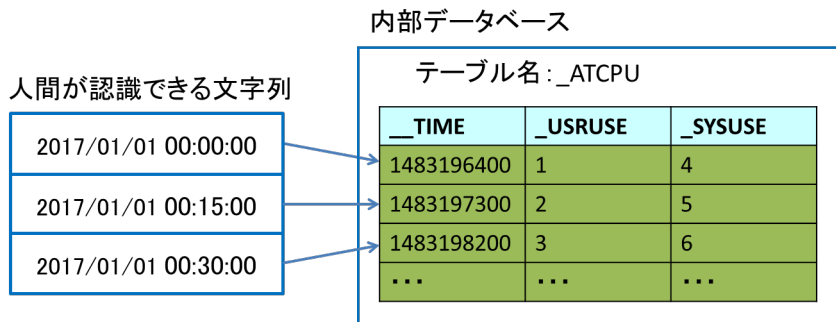
db オブジェクトは内部データベースのラッパーです。内部データベースには処理対象としているサイト／システムのパフォーマンスデータが格納されています。db オブジェクトの属性を利用することで、内部データベースの情報を確認することができます。代表的なものを紹介します。

- ✓ db.site: 処理対象としているサイトの名前 (unicode)
- ✓ db.sys: 処理対象としているシステムの名前 (unicode)
- ✓ db.evalname: 実行中のシステム評価条件ファイルの名前 (unicode)
- ✓ db.stime: 処理対象期間-開始時間 (int)
- ✓ db.etime: 処理対象期間-終了時間 (int)

1.8. 時間の扱い

内部データベースの各テーブルに存在する__TIME 列(アンダースコア 2 つ)は時間を表しています。

通常、人間が認識できる時間表現は「2017/01/01 00:00:00」などですが、__TIME 列に格納されている値は 1970 年 1 月 1 日からの経過秒数(UNIX 時間)です。



UNIX 時間を人間が認識できる文字列に変換する場合は、Python の time や datetime ライブラリを使用します。

```
import time
from datetime import datetime

def main(context, db, param):
    sql = "select __TIME from _ATCPU"
    for row in db.getcursor(sql):
        _time = row[0]
        _time_s = time.strftime("%Y/%m/%d %H:%M:%S", time.localtime(_time))
        _time_s = datetime.fromtimestamp(_time).strftime("%Y/%m/%d %H:%M:%S")
```

※ここでは、time、datetime それぞれを用いて変換していますが、通常はどちらかのライブラリを使用します。

1.9. 欠損値の扱い

CSSI は正常値を「0 以上」、欠損値を「-1」として扱います。拡張モジュールを作成する際、欠損値に注意してください。「-1」を対象に四則演算を行うと意図しない結果が返ってしまいます。必ず CSSI に用意されている各関数を使い、欠損値を計算結果に含めないようにします。以下では、_USRUSE と _SYSUSE の合計を計算しています。

SQL(_plus)による加算

```
def main(context, db, param):
    sql = "select _plus(_USRUSE, _SYSUSE) from _ATCPU"
    for row in db.getcursor(sql):
        cpuuse = row[0]
```

SQL 以外(es1_plus)による加算

```
def main(context, db, param):
    sql = "select _USRUSE, _SYSUSE from _ATCPU"
    for row in db.getcursor(sql):
        cpuuse = es1_plus(list(row))
```

1.10. チューニングヒントの作成

チューニングヒントはパフォーマンス上の様々な問題点を指摘します。重要度とメッセージの 2 つを使って表現します。重要度は 1 ～ 5 のいずれかであり、数字が小さいほど重要度が高いことになります。今回は、以下のチューニングヒントを作成します。

- ✓ 重要度：1
- ✓ メッセージ：<サイト名>/<システム名> CPU 使用率が高いです(<時間>, <CPU 使用率>%)

CSSI のチューニングヒントは、複数のチューニングヒントを資源毎にまとめるためのグループとチューニングヒントで構成され、グループにチューニングヒントを 1 つずつ追加する形になります。



まず、グループを作成します。今回は CPU の情報を扱うため、資源名を“CPU”としています。

```
def main(context, db, param):  
    group = es1_get_msg_resource("CPU", "x_CPU")
```

次に、チューニングヒントを追加します。

```
def main(context, db, param):  
    group = es1_get_msg_resource("CPU", "x_CPU")  
    group.addTuningHintMsg(1, u"CPU 使用率が高いです")
```

この拡張モジュールを実行し、評価結果閲覧画面にチューニングヒントが出力されることを確認します。

```
----- チューニングヒント -----  
重要度 1 - CPU 使用率が高いです
```

パフォーマンスデータにアクセスし、目的とするチューニングヒントを作成します。閾値は 50%にしています。

```
def main(context, db, param):  
    group = es1_get_msg_resource("CPU", "x_CPU")  
    sql = "select __TIME, _plus(_USRUSE, _SYSUSE) from _ATCPU"  
  
    for row in db.getcursor(sql):  
        _time = time.strftime("%Y/%m/%d %H:%M", time.localtime(row[0]))  
        cpuuse = row[1]  
  
        if cpuuse > 50:  
            msg = u"{}/{} CPU 使用率が高いです({}, {})%".format(  
                db.site, db.sys, _time, cpuuse)  
            group.addTuningHintMsg(1, msg)
```

拡張モジュールを実行し、評価結果を確認します。

----- チューニングヒント -----

重要度 1 - サイト/システム CPU 使用率が高いです(2017/01/01 12:00, 50.2%)

重要度 1 - サイト/システム CPU 使用率が高いです(2017/01/01 13:30, 80.0%)

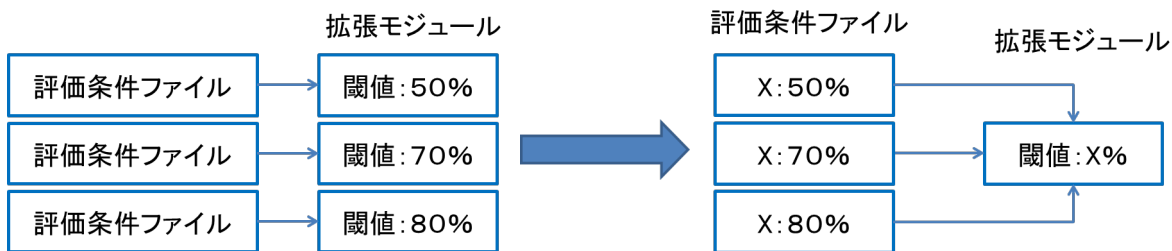
重要度 1 - サイト/システム CPU 使用率が高いです(2017/01/01 14:45, 61.0%)

...

1.11. パラメータの活用

パラメータは評価毎に拡張モジュール内の値を動的に変化させることを目的として使用します。

「チューニングヒントの作成」では 50%を閾値とした拡張モジュールを作成しましたが、異なる閾値(70%、80%など)を採用するために拡張モジュールを複製することは煩雑です。



パラメータを利用することで拡張モジュールを複製しなくても閾値を変更することができます。

パラメータは `proc_param` 属性を使います。

```
...
proc_param = {"cpu_limit": (int, 50, u"CPU 使用率の閾値", None)}
...
def main(context, db, param):
    pass
```

評価条件ファイルを追加／編集する際、`proc_param` 属性に指定した内容が GUI に表示されます。

「値」列の「50」を変更することで閾値を設定することができます。



- ✓ パラメータ : `cpu_limit`
- ✓ 値 : 50
- ✓ 説明 : CPU 使用率の閾値

GUI で設定した閾値は、main 関数の第 3 引数(param)から取得します。

```
...
proc_param = {"cpu_limit": (int, 50, u"CPU 使用率の閾値", None)}
...
def main(context, db, param):
    cpu_limit = param["cpu_limit"]
```

「チューニングヒントの作成」の拡張モジュールと組み合わせると以下ようになります。

```
...
proc_param = {"cpu_limit": (int, 50, u"CPU 使用率の閾値", None)}
...
def main(context, db, param):
    cpu_limit = param["cpu_limit"]
    group = es1_get_msg_resource("CPU", "x_CPU")
    sql = "select __TIME, _plus(_USRUSE, _SYSUSE) from _ATCPU"

    for row in db.getcursor(sql):
        _time = time.strftime("%Y/%m/%d %H:%M", time.localtime(row[0]))
        cpuuse = row[1]

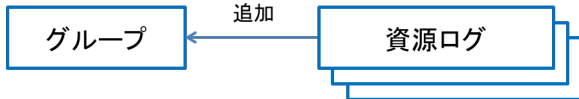
        if cpuuse > cpu_limit:
            msg = u"{} / {} CPU 使用率が高いです({}, {}%)".format(
                db.site, db.sys, _time, cpuuse)
            group.addTuningHintMsg(1, msg)
```

1.12. 資源ログの作成

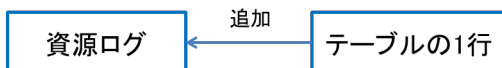
資源ログは数値や文字列を帳票形式で出力します。今回は、以下の資源ログを作成します。

✓ 列名：時間、ユーザモード使用率、カーネルモード使用率

CSSIの資源ログは、複数の資源ログをまとめるためのグループと資源ログで構成され、グループに資源ログを1つずつ追加します。



1つの資源ログは列名が定義されたひな形であり、1行ずつデータを追加します。



まず、グループを作成します。今回はCPUの情報を扱うため、資源名を「CPU」としています。

```
def main(context, db, param):  
    group = es1_get_log_resource("CPU", "x_CPU")
```

次に、資源ログのひな形を追加します。

```
def main(context, db, param):  
    group = es1_get_msg_resource("CPU", "x_CPU")  
    addlog = group.addLog(  
        u"CPU 使用率", (u"時間", u"ユーザモード使用率", u"カーネルモード使用率"))
```

addLogの「CPU 使用率」がこの資源ログの表示名、「時間」以降が列名になります。

最後に、作成したひな形に1行追加します。値は文字列のタプルです。

```
def main(context, db, param):  
    group = es1_get_log_resource("CPU", "x_CPU")  
    addlog = group.addLog(  
        u"CPU 使用率", (u"時間", u"ユーザモード使用率", u"カーネルモード使用率"))  
    addlog.addTableData(("2017/01/01 00:00", "10", "20"))
```

拡張モジュールを実行し、評価結果閲覧画面に資源ログが出力されることを確認します。

CPU 使用率		
時間	ユーザモード使用率	カーネルモード使用率
2017/01/01 00:00	10	20

パフォーマンスデータにアクセスし、目的とする資源ログを作成します。

```
def main(context, db, param):
    group = es1_get_log_resource("CPU", "x_CPU")
    addlog = group.addLog(
        u"CPU 使用率", (u"時間", u"ユーザモード使用率", u"カーネルモード使用率"))
    sql = "select __TIME, _USRUSE, _SYSUSE from _ATCPU"

    for row in db.getcursor(sql):
        _time = time.strftime("%Y/%m/%d %H:%M", time.localtime(row[0]))
        usruse = row[1]
        sysuse = row[2]

        addlog.addTableData((_time, "{}".format(usruse), "{}".format(sysuse)))
```

拡張モジュールを実行し、評価結果を確認します。

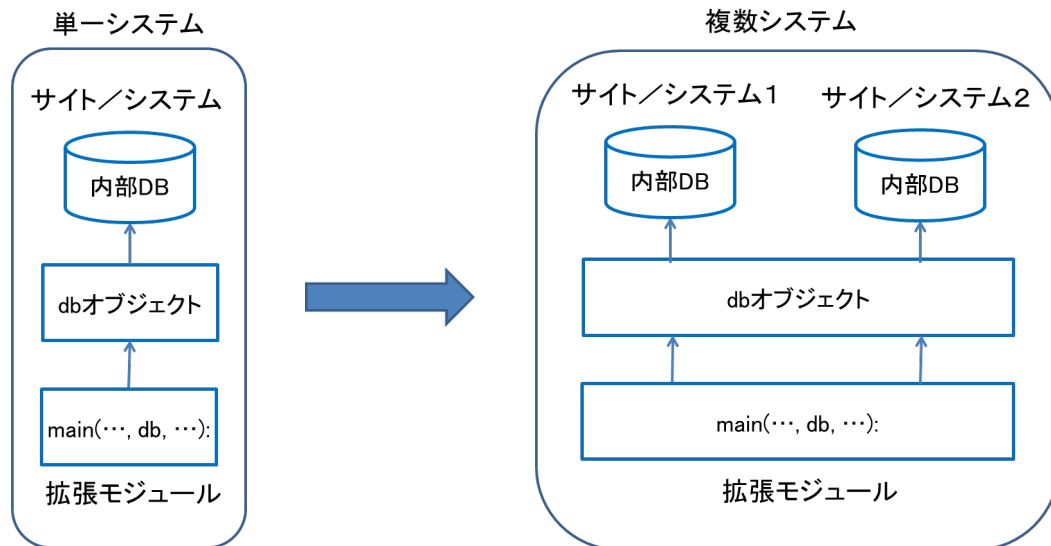
CPU 使用率		
時間	ユーザモード使用率	カーネルモード使用率
2017/01/01 00:00	15.1	35.1
2017/01/01 00:15	23.1	9.1
2017/01/01 00:30	9.0	11.0

1.13. 複数システム

前項まで一つのサイト／システム(以降、単一システム)を対象とした拡張モジュールを作成しました。
本項では複数のサイト／システムを対象とした拡張モジュールを作成します。

単一システムと複数システムの最も大きな違いは main 関数が複数回実行されることです。

- ✓ 単一システム：1 回
- ✓ 複数システム：N 回(対象システム数)



db オブジェクトは main 関数実行時点で対象としているサイト／システムのパフォーマンスデータに対応しています。

「サイト／システム 1」が対象の場合

```
def main(context, db, param):
    db.site #サイト
    db.sys #システム 1
```

「サイト／システム 2」が対象の場合

```
def main(context, db, param):
    db.site #サイト
    db.sys #システム 2
```

複数システムの拡張モジュールを作成します。まずは「何もしない」空の拡張モジュールです。

```
# -*- coding: utf-8 -*-
from es1lib import *

proc_desc = u'multi'
proc_type = 2
proc_std = False
proc_custom = True
proc_reqsym = []
proc_loadsym = []

def main_init(context, db, param):
    pass

def main(context, db, param):
    pass

def main_term(context, db, param):
    pass
```

main 関数の他に 2 つの関数(main_init、main_term)が追加されています。追加関数については「複数システム (main_init/main_term)」項で取り扱います。

評価条件ファイルを作成します。「評価対象選択」「複数システム評価(ユーザ定義)」に作成した拡張モジュール(multi)が表示されていることを確認します。



「サイト/システム選択」から複数のサイト/システムを選択後、評価を実行し、空の結果が作成されることを確認します。



「チューニングヒントの作成」項で作成した拡張モジュールから main 関数を移植します。

```
# -*- coding: utf-8 -*-
from es1lib import *
import time

proc_desc = u'multi'
proc_type = 2
proc_std = False
proc_custom = True
proc_reqsym = []
proc_loadsym = []

def main_init(context, db, param):
    pass

def main(context, db, param):
    group = es1_get_msg_resource("CPU", "x_CPU")
    sql = "select __TIME, _plus(_USRUSE, _SYSUSE) from _ATCPU"

    for row in db.getcursor(sql):
        _time = time.strftime("%Y/%m/%d %H:%M", time.localtime(row[0]))
        cpuuse = row[1]

        if cpuuse > 50:
            msg = u"{} / {} CPU 使用率が高いです({}, {}%)".format(
                db.site, db.sys, _time, cpuuse)
            group.addTuningHintMsg(1, msg)

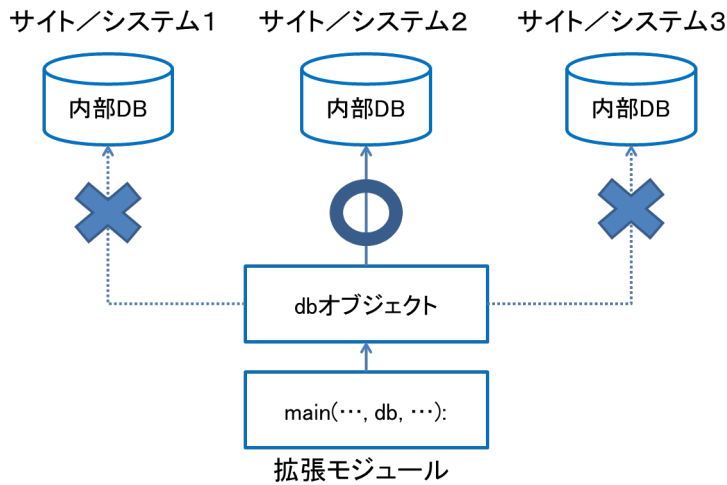
def main_term(context, db, param):
    pass
```

拡張モジュールを実行し、複数のサイト／システムのチューニングヒントが作成されていることを確認します。

```
----- チューニングヒント -----
重要度 1 - サイト/システム 1 CPU 使用率が高いです(2017/01/01 00:00, 50.2%)
重要度 1 - サイト/システム 1 CPU 使用率が高いです(2017/01/01 00:30, 80.0%)
...
重要度 1 - サイト/システム 2 CPU 使用率が高いです(2017/01/01 00:00, 70.1%)
重要度 1 - サイト/システム 2 CPU 使用率が高いです(2017/01/01 00:30, 52.3%)
...
重要度 1 - サイト/システム 3 CPU 使用率が高いです(2017/01/01 04:30, 57.0%)
重要度 1 - サイト/システム 3 CPU 使用率が高いです(2017/01/01 05:00, 77.0%)
```

1.14. 複数システム(main_init/main_term)

「複数システム」項に記載の通り、db オブジェクトは対象サイト／システムが切り替わるたび、更新されます。言い換えると「main 関数から参照可能なパフォーマンスデータは処理中のサイト／システムのみ」となります。以前、以降のサイト／システムを参照することはできません。



複数サイト／システムのデータを一括して取り扱う場合、main 関数実行時に取得したデータを拡張モジュール内に保持し、全ての main 関数が終わった後に保持したデータを参照することになります。

拡張モジュール内でデータを保持するためのオブジェクトが context オブジェクトです。context オブジェクトは main_init/main/main_term 関数の第 1 引数です。

```
def main(context, db, param):  
    pass
```

main_init 関数は main 関数実行前に 1 度だけ呼び出される関数です。後続の main 関数のデータを保持するための属性 (data) を context オブジェクトに追加します。

```
def main_init(context, db, param):  
    context.data = []
```

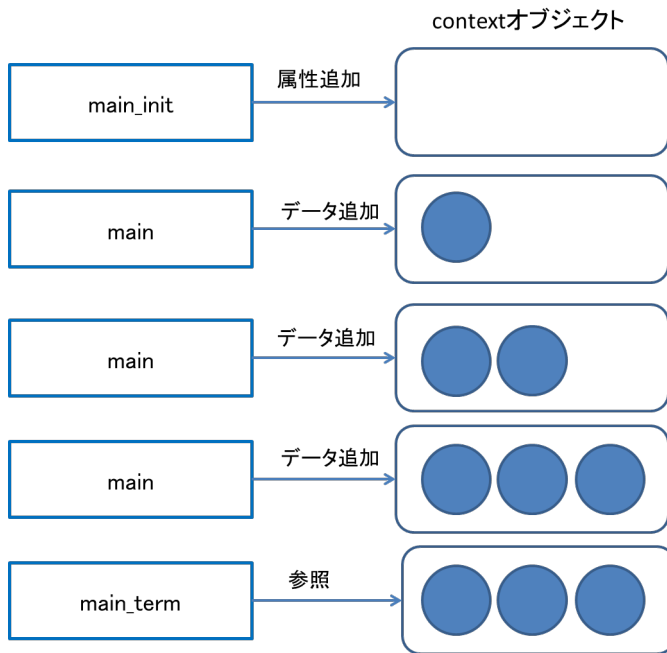
main 関数ではサイト／システムのデータを context.data オブジェクトに追加します。

```
def main(context, db, param):  
    context.data.append(...)
```

main_term 関数は全ての main 関数実行後に 1 度だけ呼び出される関数です。context.data オブジェクトに格納された全サイト／システムのデータを取得し処理を行います。

```
def main_term(context, db, param):  
    for v in context.data:  
        # do something
```

図に表すと以下のような形で関数を跨ったデータの引き継ぎを行います。



以上を踏まえ、複数サイト／システムのデータを一括で扱う以下の資源ログを作成します。
「時間」は最大値を記録した時間です。

✓ 列名：CPU 使用率(最大)、時間

サイト／システム	CPU使用率(最大)	時間
サイト／システム1	99.9	2017/01/01 13:45
サイト／システム2	82.0	2017/01/01 15:00
サイト／システム3	73.4	2017/01/01 18:15
...

まず、main_init 関数を作成します。

追加した context.data オブジェクトに資源ログの各列に対応した値を格納する予定です。

```
def main_init(context, db, param):
    context.data = []
    # [[sitesys, cpu_max, _time], ]
```

次に、main 関数を作成します。パフォーマンスデータにアクセスし、context.data オブジェクトにデータを追加します。

```
def main(context, db, param):
    sql = "select __TIME, _plus(_USRUSE, _SYSUSE) from _ATCPU order by 2 desc limit 1"
    for row in db.getcursor(sql):
        _time = time.strftime("%Y/%m/%d %H:%M", time.localtime(row[0]))
        cpu_max = row[1]
        data = [
            u"{}/{}".format(db.site, db.sys), "{}".format(cpu_max), "{}".format(_time)]
        context.data.append(data)
```

最後に、main_term 関数を作成します。

context.data オブジェクトから各サイト／システムのデータを取得し、資源ログに出力します。

```
def main_term(context, db, param):
    group = es1_get_log_resource("CPU", "x_CPU")
    addlog = group.addLog(u"CPU 使用率", (u"サイト／システム", u"CPU 使用率(最大)", u"時間"))
    for row in context.data:
        addlog.addTableData(row)
```

拡張モジュールを実行し、評価結果を確認します。

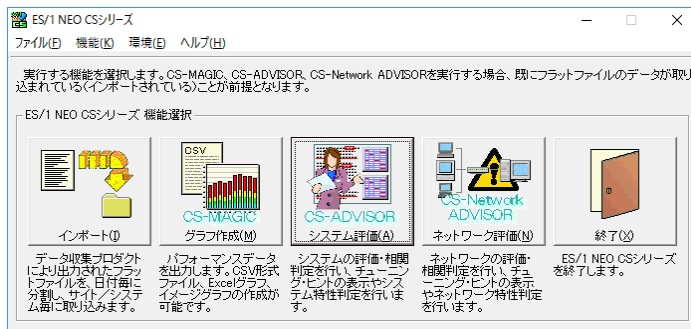
CPU 使用率			
サイト／システム	CPU 使用率(最大)	時間	
サイト/システム 1	96.0	2017/01/01 10:15	
サイト/システム 2	92.0	2017/01/01 11:30	
サイト/システム 3	98.0	2017/01/01 04:00	

1.15. エラーへの対処

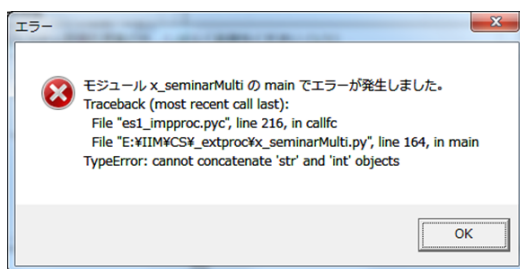
CSSI エンジン拡張モジュールを import します。import に失敗すると GUI に表示されない、実行できないなどの問題が発生します。import 結果は以下のファイルに出力されます。作成した拡張モジュールが正しく import されていることを確認してください。

- ✓ IIM インストールフォルダ¥CS¥pyimport.txt

import は「ES/1 NEO CS シリーズ」画面の「システム評価」押下時に行われます。
新規に拡張モジュールを作成した場合は、該当画面に戻り import を行ってください。



GUI から拡張モジュールを実行した場合、エラーメッセージがダイアログボックスに表示されます。



バッチから拡張モジュールを実行した場合、以下のファイルにエラーメッセージが出力されます。

- ✓ IIM インストールフォルダ¥CS¥csadvcr.log
- ✓ IIM インストールフォルダ¥CS¥csexecpy_YYYYMMDD_HHMMSS_xxx.dmp

1.16. 実行可能なサンプルスクリプト

下記のフォルダに実行可能なサンプルスクリプトを用意しています。

[フォルダ]

(ES/1 NEO CS シリーズ インストールフォルダ) ¥CS¥sampleproc

[スクリプトファイル]

- x_SystemResourceCheck.py (システムリソース評価)
- x_LoopCheck.py (プロセスループ検出)
- x_MonthlyCompare.py (前月比較)
- x_Interval Summry. py (インターバルサマリー)
- x_orclmonth.py (Oracle 月次レポート)
- x_Ec2InstanceTypeCalc.py (EC2 インスタンス)

[利用方法]

下記のフォルダに使用するスクリプトファイルをコピーしてください。

(ES/1 NEO CS シリーズ インストールフォルダ) ¥CS¥_extproc

(*)x_orclmonth.py をコピーする場合は orbase.py も一緒にコピーして下さい。

コピーを行うと CS-ADVISOR 評価条件ファイルウィザードの評価対象選択に表示されます。

- | | |
|-----------------------------------|--------------------|
| •システムリソース評価 | → 単体システム評価 (ユーザ定義) |
| •プロセスループ検出 | → 複数システム評価 (ユーザ定義) |
| •前月比較 | → 複数システム評価 (ユーザ定義) |
| •インターバルサマリー | → 複数システム評価 (ユーザ定義) |
| •Oracle 月次レポート(*) | → 単体システム評価 (ユーザ定義) |
| •Amazon Web Services – EC2 インスタンス | → 単体システム評価 (ユーザ定義) |

(*)画面上には"Oracle(月次)"と表示されます。

システムリソース評価

[機能]

単一のシステムからプロセッサ、メモリ、ディスクのパフォーマンスデータを読み込み、使用状況に問題がある資源項目を検出します。評価結果は資源ログに出力します。

- ・プロセッサ使用率
- ・メモリ使用率
- ・ディスク使用率
- ・ファイルスペース使用率

[判定]

各資源項目の使用率が高すぎる場合と低すぎる場合を問題ありとして判定します。高低の判断基準には、資源毎に 5 段階の任意の値を指定できます。閾値と比較を行う統計値には、デフォルトとして平均値を採用しています。平均値以外にも次の統計値を比較対象とすることができます。

- ・平均値
- ・合計値
- ・最小値
- ・最大値
- ・n パーセンタイル値

[環境設定パラメータ]

メモリの評価を行う場合は、環境設定パラメータの設定が必要です。評価条件ファイルを登録する際に、CS-ADVISOR の GUI からメモリ搭載量(MB)およびページサイズ(KB)を設定してください。

プロセスループ検出(x_LoopCheck.py)

[機能]

複数のシステムからプロセスがループしているシステムを検出しチューニングヒントに出力します。

[判定]

プロセッサ使用率のデータを最新の時刻から読み込みます。100%をプロセッサ台数で除算した値の倍数をループ判定の閾値とします。例えば、プロセッサ台数が 4 台の場合「100% / 4 = 25%, 50%, 75%, 100%」がループ判定の閾値となります。閾値には、±αの範囲を設定することが可能です。変数名 LOOP_RANGE がそれに該当し、デフォルト値は 8.0 となっています。±αの範囲は、(LOOP_RANGE の値 / プロセッサ台数)となります。

また、閾値の範囲に連続でプロセッサ使用率が存在した回数を設定することが可能です。変数名 LOOP_COUNT がそれに該当し、デフォルト値は 8 となっています。つまり、8 回連続で閾値の範囲に該当した場合、そのシステムはプロセスループが発生していると判定します。

[環境設定パラメータ]

評価処理で使用する下記の 2 つのパラメータを CS-ADVISOR の GUI から指定することが可能です。デフォルト値を変更する場合は、それぞれ値を設定してください。

パラメータ	値	説明
LOOP_COUNT	8	プロセスループ判別のチェック回数
LOOP_RANGE	8.0	プロセスループ判別の許容範囲(%)

前月比較(x_MonthlyCompare.py)

[機能]

複数のシステムから 2 ヶ月分のデータを読み込み、前月との比較を行います。比較結果は資源ログに出力します。

- ・プロセッサ使用率
- ・フリーメモリ
- ・ディスク使用率

[判定]

前月と比ぶ当月の平均値の増分が閾値を超えている場合はチューニングヒントを出力します。評価に使用する閾値は変数となっており、変更することが可能です。

- ・プロセッサ使用率 : CHECK_CPU
- ・フリーメモリ : CHECK_MEMORY
- ・ファイルスペース : CHECK_DISK

[環境設定パラメータ]

評価処理で使用する下記の 3 つのパラメータを CS-ADVISOR の GUI から指定することが可能です。

デフォルト値を変更する場合は、それぞれ値を設定してください。

パラメータ	値	説明
CHECK_CPU	30	プロセッサ使用率(%)の増分の閾値
CHECK_MEMORY	100	フリーメモリ(MB)の増分の閾値
CHECK_DISK	30	ファイルスペース使用率(%)の増分の閾値

インターバルサマリー (x_IntervalSummary.py)

[機能]

複数のシステムから WEB サーバ、AP サーバ、DB サーバの種別を判定し、各システムのインターバルサマリーを資源ログに出力します。

[判定]

CPU 使用率が閾値を超えた場合を問題ありとして判定します。

[環境設定パラメータ]

評価処理で使用する下記の 5 つのパラメータを CS-ADVISOR の GUI から指定することが可能です。

デフォルト値を変更する場合は、それぞれ値を設定してください。

パラメータ	値	説明
CPU (WEB サーバ)	80	WEB サーバの CPU 使用率の閾値(%)
CPU (AP サーバ)	80	AP サーバの CPU 使用率の閾値(%)
CPU (DB サーバ)	80	DB サーバの CPU 使用率の閾値(%)
CPU (その他サーバ)	80	その他サーバの CPU 使用率の閾値(%)
NonDisplay (その他サーバ)	はい	その他サーバを非表示にする

Oracle 月次レポート(x_orclmonth.py)

[機能]

1 ヶ月間の Oracle データをもとにレポートを作成します。

[判定]

このスクリプトにより出力される内容の説明は「システム・チューニングガイド」を参照してください。

[環境設定パラメータ]

必要なし

EC2 インスタンス(x_Ec2InstanceTypeCalc.py)

[機能]

Amazon Web Services の EC2 インスタンスの稼働データを読み込み、過負荷な EC2 インスタンス、低負荷な EC2 インスタンスを検出します。

[判定]

CPU 使用率が閾値を超えた場合を検出して判定します。評価に使用する閾値は変数となっており、変更することが可能です。

[環境設定パラメータ]

評価処理で使用する下記の 3 つのパラメータを CS-ADVISOR の GUI から指定することが可能です。
デフォルト値を変更する場合は、それぞれ値を設定してください。

パラメータ	値	説明
UndersizedInstanceUpperLimitForCpuUsage	95	高負荷インスタンス判定の CPU 使用率上限値
UndersizedInstanceUpperLimitExceededIntervals	3	高負荷インスタンス判定の連続インターバル数
OversizedInstanceLowerLimitForCpuUsage	15	低負荷インスタンス判定の CPU 使用率下限値

第2章 CS Scripting Interface リファレンス

2.1. はじめに

この文書は CS Scripting Interface（以降"CSSI"と表記します）で使用する CSSI 独自の Python(*1)オブジェクトについてのリファレンスです。CSSI の概略については「第 1 章 CS Scripting Interface チュートリアル」を参照してください。

以降の文中で手続き（関数やメソッド）の宣言を、

手続き名(引数...)

のように記述します。引数で"x=y"のように記述されている場合はその引数(x)にはデフォルト値(y)があり省略可能であることを示します。例えば

fc(a, b, c=100)

のように宣言されている手続き fc については呼び出し fc(1, 2)は fc(1, 2, 100)と同じ意味になります。

メモ！

(*1) 公式ホームページは <http://www.python.org/> です。CSSI が使用している Python のバージョンは 2.7 です。

2.2. 拡張モジュール

拡張モジュールは、ES/1 NEO CS シリーズが蓄積したパフォーマンスデータを評価するためのモジュールです。拡張モジュールに任意の評価ロジックを記述することで、ユーザ固有の評価を行うことができます。

2.2.1. パフォーマンスデータの取り扱い

CSSI では各種のパフォーマンスデータを大きく「表」という単位に分類し、表中に分類された個々のパフォーマンスデータを「列」という項目に対応させて取り扱います。表は固有の名前で識別され、列はそれが属する表中での固有の名前で識別されます。例えば"ATCPU"という名前の表にはプロセッサに関連する情報が対応し、"ATCPU"表にはユーザモードでのプロセッサ使用率を示す"USRUSE"という名前の列やカーネルモードでのプロセッサ使用率を示す"SYSUSE"という名前の列が存在します。また、表中に存在するパフォーマンスデータの実体は「レコード」、レコードの列に相当する部分は「フィールド」として取り扱います。

拡張モジュールで"from es1lib import "と記述することにより、表（またはレコード）をあらわすオブジェクト（レコードオブジェクト）と、列（またはフィールド）をあらわすオブジェクト（フィールドオブジェクト）を使用することが可能です。レコードオブジェクトはそれに対応する表と同一の名前を持つ（変数にセットされた）オブジェクトとしてアクセスできます。例えばプロセッサ情報の表の名前は"ATCPU"なので、これに対応するレコードオブジェクトは拡張モジュール中で ATCPU としてアクセスできます。フィールドオブジェクトはそれが属するレコードオブジェクトの属性としてアクセス可能であり、属性名は対応する列と同一の名前になります。例えばプロセッサ情報のユーザモード使用率を示す列の名前は"USRUSE"なので、これに対応するフィールドオブジェクトは拡張モジュール中で ATCPU.USRUSE としてアクセスできます(*2)。

使用可能な表名と列名については ES/1 NEO CS シリーズのマニュアル「CS-MAGIC 使用者の手引き」中の「第 9 章 ES/1 NEO CS シリーズのクエリーで使用可能なデータ列名」を参照してください(*3)。

メモ！

(*2) Python では変数名や属性名を、英大文字小文字を区別して取り扱います。従って上記の ATCPU や USRUSE といった綴りはすべて大文字で記述してください。

(*3) 一部例外があります。例外については後述の「2.8. レコード/フィールドオブジェクトの例外規定」を参照してください。

2.2.2. 拡張モジュールの名前

拡張モジュールは CS インストールフォルダ以下の "_extproc" フォルダ内に作成します。

拡張モジュールのファイル名は、"x_" のプレフィックスで始まり、".py" の拡張子で終わり、"x_" と拡張子の間は英数字にて構成されている必要があります。

2.2.3. 拡張モジュールでの日本語の使用

拡張モジュールに日本語文字（列）を含める場合は拡張モジュールの保存ファイルの文字エンコード形式を utf-8 とし、拡張モジュールの 1 行目か 2 行目にエンコーディングが utf-8 であることを示すコメント行(*4)を記述します。また処理中の日本語を含む文字（列）の先頭には'u'を付加しそれが unicode 文字列であることを明示してください。

メモ！

(*4) 1、2 行目に記述するエンコーディングの指定は以下の正規表現にマッチする文字列を含んでいれば下記例の通りでなくても構いません。

```
coding[ : = ]¥s*[uU][tT][fF]-8
```

(例)

```
# coding=utf-8
```

```
MSGTEXT=u"日本語メッセージテキスト"
```

2.2.4. 拡張モジュールの属性

拡張モジュールは幾つかのモジュール属性を持つ必要があります。これらの属性は評価条件ファイル作成時や評価（拡張モジュール）実行時に参照されます。以下に拡張モジュールに必要な属性の一覧を示します。"（デフォルト：...）"となっている属性については省略可能であり、省略時には"... "で示した値が使用されます。"（省略可）"となっている属性は省略可でありオプション的な意味を持つ（省略した場合は何もしない）属性です。"（省略不可）"となっている属性は必ず指定するようにしてください（指定が無い場合は拡張モジュールとしては扱われません）。

proc_type：（省略不可／整数型）

拡張モジュールが単一システムの評価実行時に使用するものならば 1 を、複数システムの評価実行時に使用するものならば 2 を指定してください。

```
proc_type=1    # 単一システム用
```

メモ！

出力成果物は次の表のように異なります。目的に応じた設定を行ってください。

	PWS 連携	テキスト出力	Excel 出力	GIF/PNG 出力	CSV 出力
単体システム評価	○	○	○	○	○
単体システム評価 （ユーザ定義）	○	○	○	○	×
複数システム評価 （ユーザ定義）	○	○	○	○	×

proc_std : (デフォルト : False / 論理型)

CS 標準提供処理と同時に実行可能ならば True をそうでなければ False を指定してください。

	単体システム評価	単体システム評価 (ユーザ定義)	複数システム評価 (ユーザ定義)
True	○	○	○
False	×	○	○

```
proc_std=True    # CS 標準提供処理と同時に実行可能
```

proc_custom : (デフォルト : True / 論理型)

CS 標準提供処理とは別に実行可能ならば True をそうでなければ False を指定してください。

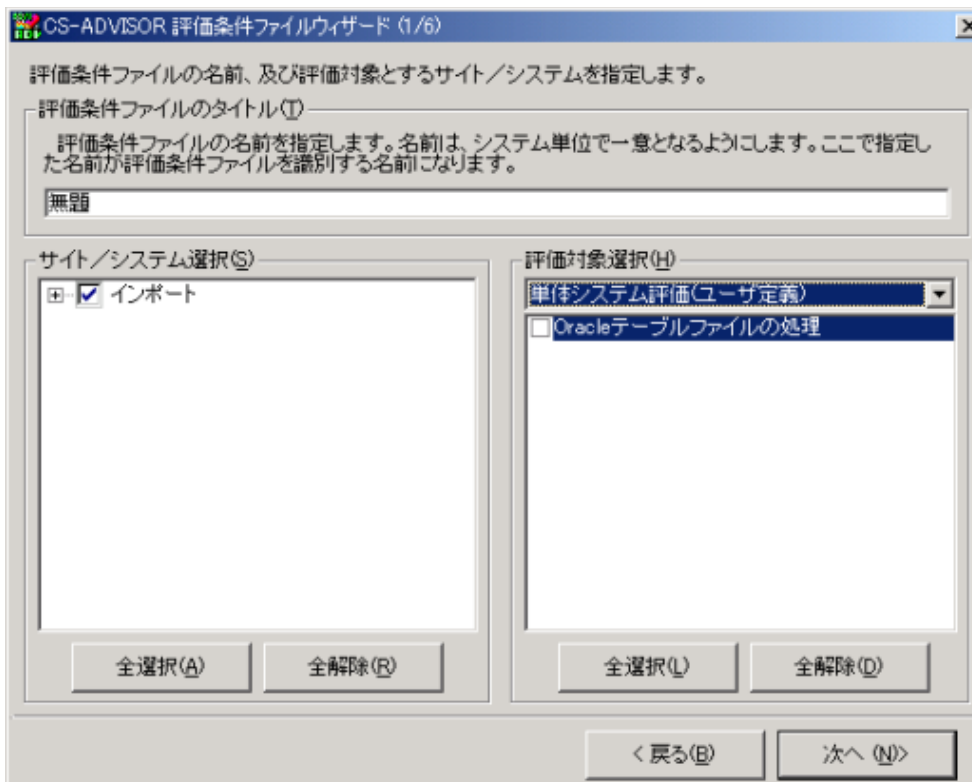
	単体システム評価	単体システム評価 (ユーザ定義)	複数システム評価 (ユーザ定義)
True	proc_std=True の場合は、○	○	○
False	proc_std= False の場合は、×	×	×

```
proc_custom=True    # 独立して実行可能
```

proc_desc : (省略不可 / 文字列型)

評価条件作成の GUI にてこの拡張モジュールの選択 (識別) 文字列として表示する文字列を指定してください。

```
proc_desc=u"Oracle テーブルファイルの処理"
```



proc_param : (デフォルト : { } / ディクショナリ型)

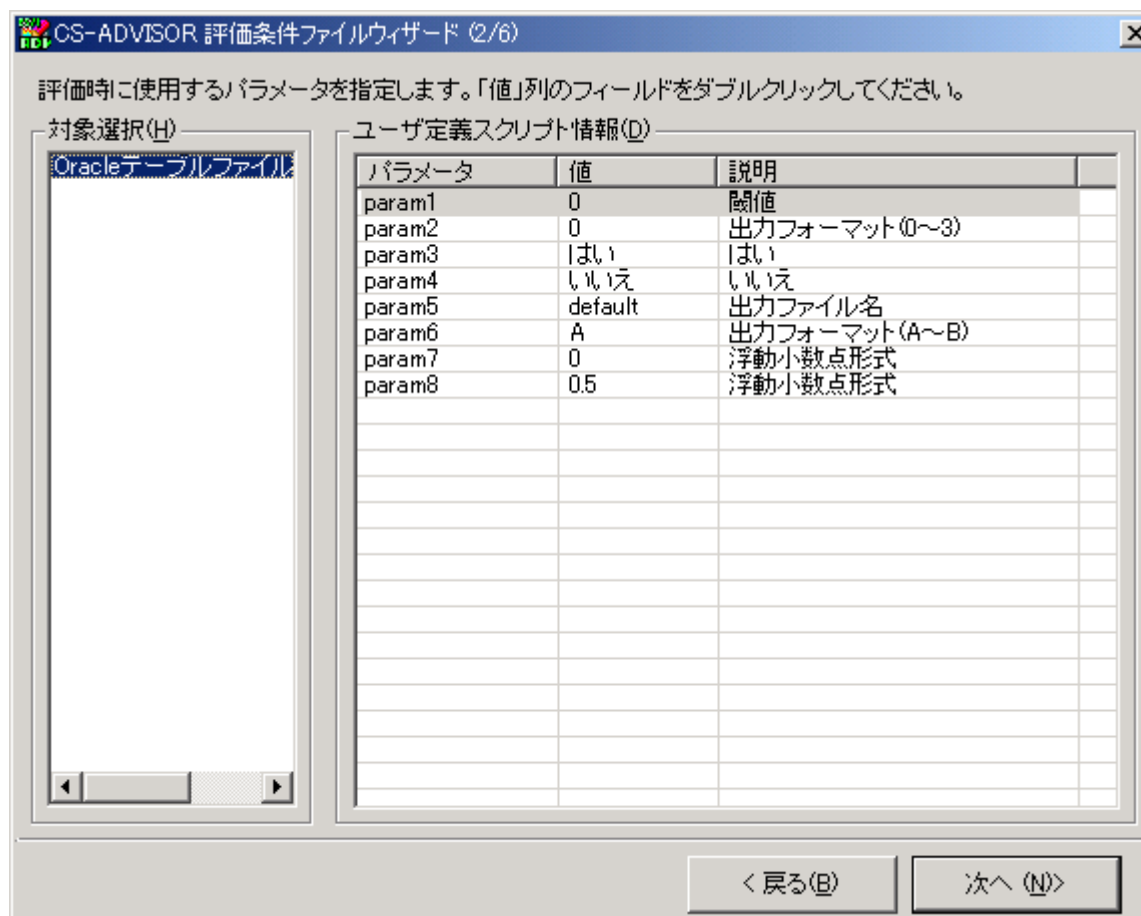
拡張モジュールが実行時にパラメータとして受け取る情報の定義を、パラメータ名をキーとしたディクショナリの形で記述してください。キーに対応する値には、(パラメータの型, デフォルト値, パラメータの説明, 値の候補) のタプル (またはリスト) を指定してください。

パラメータ名にはアルファベットで始まりアルファベット・数字・アンダースコア ('_') で構成される文字列を、パラメータの型には Python の型オブジェクトである unicode、str、int、float、bool のいずれかを、デフォルト値にはパラメータの型に応じた値 (デフォルトを設けない場合は None) を、パラメータの説明には文字列を、値の候補にはパラメータの型に応じた値のタプル (またはリスト) をそれぞれ指定してください。

マルチバイト文字列を扱う場合は unicode を指定してください。値の候補が存在しない場合は値の候補に None または空のシーケンスを指定してください。

この属性は評価条件作成時に (GUI より) 参照され、この属性の定義に基づく評価条件 (拡張モジュール) 実行時のパラメータを指定することが可能となります。

```
proc_param = { "param1" : (int, 0, u'閾値',      None),
               "param2" : (int, 0, u"出力フォーマット(0~3)", (0,1,2,3)),
               "param3" : (bool, True, u"はい",      None),
               "param4" : (bool, False, u"いいえ",    None),
               "param5" : (str, "default", u"出力ファイル名", None),
               "param6" : (str, "A", u"出力フォーマット(A~B)", ("","A","B")),
               "param7" : (float, 0, u"浮動小数点形式", None),
               "param8" : (float, 0.5, u"浮動小数点形式", (0.1,0.5,0.9)) }
```



proc_chkparam : (デフォルト : 常に True を返す関数)

上記 proc_param で指定したパラメータ値の妥当性チェックを行う関数を指定してください。

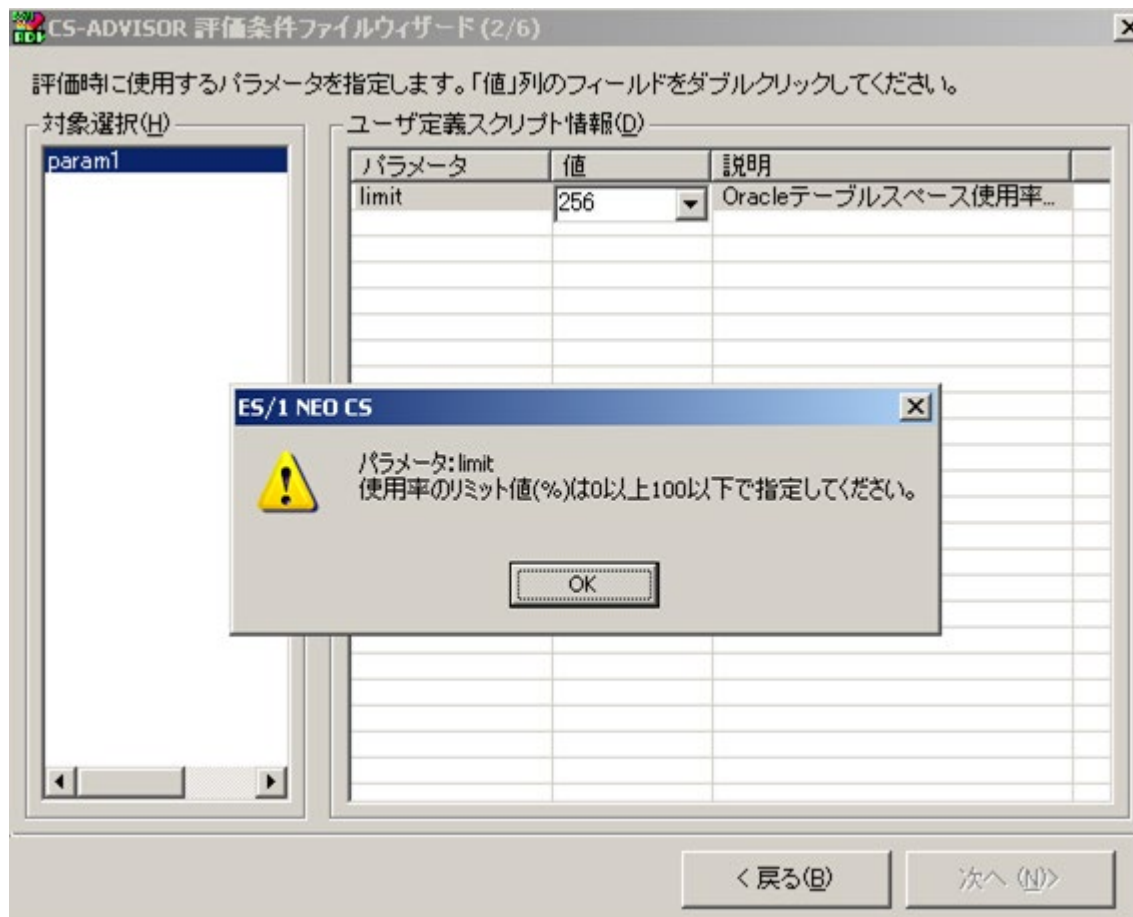
指定する関数はパラメータ名をキーとしそれに対応する値を持つディクショナリを引数として受け取らなくてはなりません。

関数の戻り値は、受け取った引数（パラメータのディクショナリ）が妥当であれば True とし、そうでなければ問題のあったパラメータ名とそのエラーメッセージのタプルとしてください。

この属性(関数)は評価条件作成時に（GUI より）呼び出され、指定されたパラメータが有効か否かを判定します。

```
proc_param = { 'limit' : (int, 90, u'Oracle テーブルスペース使用率のリミット値(%)', None) }
```

```
def proc_chkparam (param):  
    if not (0 <= param["limit"] <= 100):  
        return "limit", u"使用率のリミット値(%)は 0 以上 100 以下で指定してください."  
    return True
```



proc_reqsym : (デフォルト : [] / リスト型)

拡張モジュールの処理実行に最低限必要なレコードオブジェクトをシーケンスの形で列挙してください。評価条件（拡張モジュール）実行時にこの属性に列挙されたいずれのレコードも対象システムのデータ中に存在しない場合、この拡張モジュールの処理実行はスキップされます。

この属性が空のシーケンスの場合はレコードの存在状況に関わらずこの拡張モジュールの処理を実行することを意味します。

```
proc_reqsym = [ORFILEIO]    # Oracle データファイル情報が存在しない場合は処理をスキップ
```

proc_loadsym : (デフォルト : [] / リスト型)

拡張モジュールの処理実行に使用するレコードオブジェクトをシーケンスの形で列挙してください。この属性に列挙されなかったレコードを処理実行時に参照しようとすると例外が発生します(*5)。

この属性に空のシーケンスを指定した場合は処理実行時にすべてのレコードが参照可能となります(*6)。

```
proc_loadsym = [ATCPU, ATPAGE] # CPU/ページング情報のみを使用
```

メモ！

- (*5) CS 標準提供処理で使用するレコードや他の拡張モジュールが使用するレコードを参照した場合には例外が発生しない場合もあります。ただし、他の処理への依存を避ける意味でこの属性には明示的に必要なレコードを指定するようにしてください。
- (*6) すべてのレコードを参照可能とした場合、必要最低限のレコードを指定した場合に比較してデータのアクセスに多くの時間がかかる可能性があります。出来るだけ必要最低限のレコードを指定するようにすることをお勧めします。

proc_fftype : (デフォルト : ES1_FFTYPE_NONE / 定数より選択)

評価条件ファイル作成時に表示される「CS-ADVISOR 評価条件ファイルウィザード (4/5)」画面の「集約データの指定」欄の種別を指定します。指定可能な値は以下です。

ES1_FFTYPE_DETAIL : 未集約のフラットファイルが対象
ES1_FFTYPE_DAILY : 日毎集約機能で作成したフラットファイルが対象
ES1_FFTYPE_MONTHLY : 月毎集約機能で作成したフラットファイルが対象

明示的に指定を行わない場合はこの属性をコメントアウトしてください。

```
proc_loadsym = ES1_FFTYPE_DETAIL # 未集約のフラットファイルが対象
```

main : (省略不可)

拡張モジュールの処理本体を実行する関数を記述してください。この関数は引数として context、db、param の 3 つの引数を受け取らなくてはなりません。

- context : (省略不可／任意のオブジェクト)
複数システムでの評価実行時に、main 関数間の呼び出しをまたがって引き継ぐ必要があるオブジェクトを保持するためのオブジェクトが渡されます（下記「複数システム評価時の main/main_init/main_term 呼び出し手順」を参照してください）。
- db : (省略不可／db オブジェクト)
処理対象システムのデータが格納されたオブジェクト（詳しくは後述の「2.3. db オブジェクト」を参照してください）が渡されます。
- param : (省略不可／ディクショナリ型)
評価条件作成時に指定されたこの拡張モジュールの実行時パラメータを含む（パラメータ名と指定値が組になった）ディクショナリが渡されます。

main_init : (省略可)

複数システムの評価実行に使用する（proc_type=2 を指定した）拡張モジュールの処理本体（main 関数）実行に先立ち実行する関数を記述してください。この関数は引数として context、db、param の 3 つの引数を受け取らなくてはなりません。引数の意味は上記の main における意味と同様です。

単一システムの評価実行に使用する（proc_type=1 を指定した）拡張モジュールについてはこの関数の記述は無視されます（評価実行時に呼び出されません）。

main_term : (省略可)

複数システムの評価実行に使用する（proc_type=2 を指定した）拡張モジュールの処理本体（main 関数）実行終了後に実行する関数を記述してください。この関数は引数として context、db、param の 3 つの引数を受け取らなくてはなりません。引数の意味は上記の main における意味と同様です。

単一システムの評価実行に使用する（proc_type=1 を指定した）拡張モジュールについてはこの関数の記述は無視されます（評価実行時に呼び出されません）。

複数システム評価時の main/main_init/main_term 呼び出し手順

複数システムを対象とした評価条件中に指定された各拡張モジュールの main、main_init、main_term 関数のそれぞれの呼び出し手順は以下のようになります。

- (1)各拡張モジュールの main_init 関数の呼び出し
- (2)評価条件中に指定された対象システム毎に以下を実行
 - ①評価対象期間のデータを db オブジェクト（以降「2.3. db オブジェクト」を参照）に格納
 - ②db オブジェクトを引数に伴い、各拡張モジュールの main 関数の呼び出し
- (3)各拡張モジュールの main_term 関数の呼び出し

上記のように評価条件中に複数の対象システムを含む場合には（対象システム毎に）main 関数が複数回呼び出されることになります。main、main_init、main_term の各関数の第 1 引数である context はこのような main 関数の複数回呼び出し時に各 main 関数の呼び出し間にまたがりオブジェクトを保持するためのものです。context は単に以下のように定義されたクラスのインスタンスです。

```
class ProcContext (object) :  
    pass
```

例えば各 main 関数の実行で 1 つのファイルに情報を書き出したい場合は以下のように記述することでそれを実現できます。
次の 4 つのサイト／システムを対象に後述のソースコードを実行した例を示します。

[Site1/System1]、[Site1/System2]、[Site2/System3]、[Site2/System4]

```
import codecs  
  
def main_init (context, db, param):  
    context.F = codecs.open("C:¥¥foo.txt","w","mbcs")  
  
def main (context, db, param):  
    print >> context.F, "%s/%s" % (db.site, db.sys)  
  
def main_term (context, db, param):  
    context.F.close()
```

C:¥¥foo.txt への出力結果です。4 つのシステムが対象のため、main 関数は 4 回実行されています。
main_init と main_term は 1 回ずつの実行となります。



corr_main : (省略可)

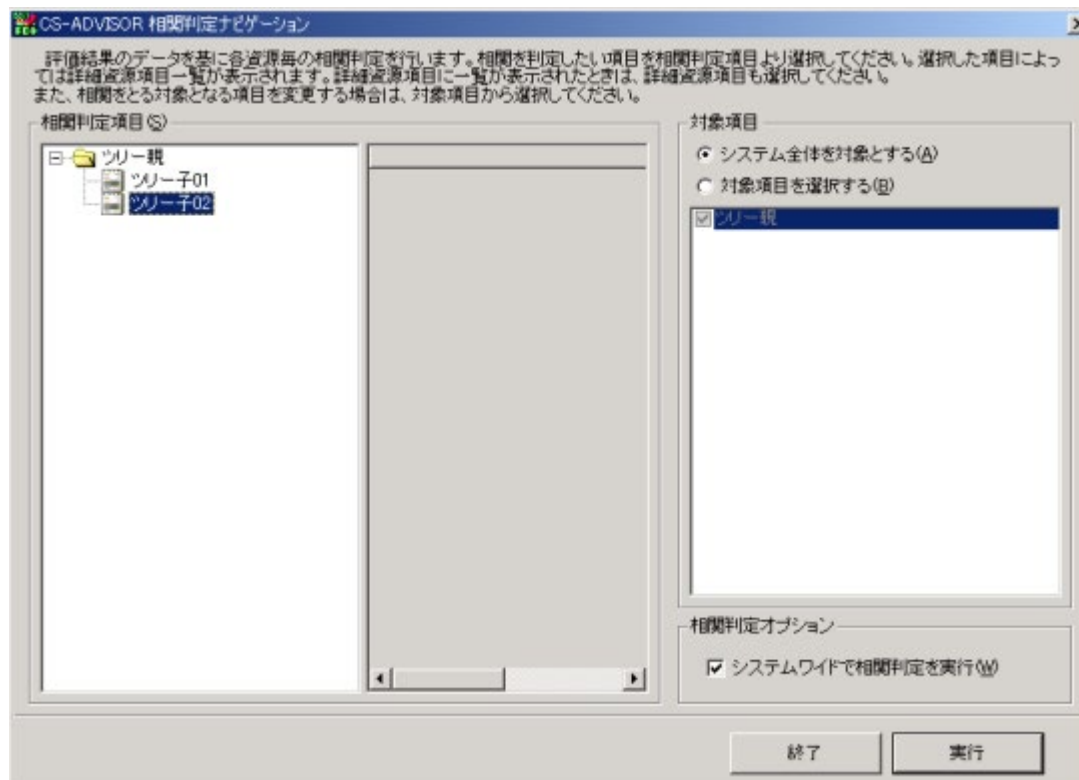
相関判定ナビゲーションとの連携を行うための関数を記述してください。この関数は引数として context、db、param の3つの引数を受け取らなくてはなりません。引数の意味は上記の main における意味と同様です。

この関数内では（この拡張モジュールを使用した評価結果から実行される）相関判定ナビゲーション処理で使用する相関判定対象項目についての表示方法と計算方法を指定できます（指定方法については以降「2.5.5. 相関判定ナビゲーションとの連携」を参照してください）。複数システムの評価実行に使用する（proc_type=2 を指定した）拡張モジュールについてはこの関数の記述は無視されます（評価実行時に呼び出されません）。

```
# ※使用している関数は後述しております。ここでは説明を割愛いたします。
def main (context, db, param):
    # ATCPU.USRUSE と ATCPU.SYSUSE で相関判定を行いメッセージを作成します
    msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
    ctproc = es1_CorrTerm(u'ユーザ使用率' , "_USRUSE" , record=ATCPU)
    ctsysc = es1_CorrTerm(u'システム使用率' , "_SYSUSE" , record=ATCPU)
    es1_add_corrmsg(ctproc, ctsysc, msgres)

def corr_main (context, db, param):
    # ATCPU.USRUSE と ATCPU.SYSUSE で相関判定ナビゲーションを行う
    ng = es1_getNaviGroup(u"ツリー親")
    ng.addNaviItem(u"ツリー子 01", "_USRUSE", ATCPU)
    ng.addNaviItem(u"ツリー子 02", "_SYSUSE", ATCPU)
```

※相関判定ナビゲーションの結果はデータに左右されるため、ここでは割愛します。



2.3. db オブジェクト

db オブジェクトはパフォーマンスデータが格納されたオブジェクトです。パフォーマンスデータは実際には db オブジェクトが保持するリレーショナルデータベース（以降このリレーショナルデータベースを"RDB"と表記します）に格納されています。

RDB 中にはパフォーマンスデータの表に相当するテーブルが存在し、各テーブルにはパフォーマンスデータの列に相当するカラムが定義されています。テーブルとカラムの名はそれに対応する表や列の名前の先頭にアンダースコア（'_'）を付けたものです。例えば表 ATCPU の列 USRUSE には、RDB のテーブル "_ATCPU" のカラム "_USRUSE" が対応します。また各テーブルにはレコードの日時を示す "__TIME" という名のカラムとレコードのインターバル長を示す "__INTVL" という名のカラムが定義されています(*7)。レコードの日時は基準時（1970/01/01 00:00:00）からの経過秒数を示す整数値で Python の time モジュールを用いて変換することが可能です。インターバル長は秒を示す整数で表現されています。

メモ！

(*7) 一部例外があります。例外については後述の「2.8. レコード/フィールドオブジェクトの例外規定」を参照してください。

2.3.1. RDB への直接アクセス

db オブジェクトは RDB への直接アクセスをサポートする以下のメソッドを持ちます。

getcursor(SQL)

SQL : (省略不可／文字列型)
RDB に対する select 文(*8)を記述した文字列を指定してください。

メソッドの戻り値は select 文の抽出結果に順次アクセスするためのオブジェクト（Cursor オブジェクト）です。Cursor オブジェクトに for を用いてアクセスすることにより抽出結果の各レコードを表現するオブジェクト（row オブジェクト）を順次得ることができます。row オブジェクトにインデックス（select 文で抽出対象とした最も左側のフィールドを 0 と数えます）を用いてアクセスすることによりレコードの特定フィールドの値を取得することが可能です。"select * ..."とした場合、フィールドの抽出順は不定となりインデックスによるフィールドの特定ができなくなりますので注意してください。

メモ！

(*8) 標準の SQL 規格（SQL92）に則った表記をサポート（RIGHT OUTER JOIN と FULL OUTER JOIN を除く）します。

```
context.F = open("C:¥¥foo.txt", "w")
for row in db.getcursor("select __TIME, __INTVL, _USRUSE, _SYSUSE from _ATCPU"):
    ts = time.strftime("%Y/%m/%d %H:%M", time.localtime(row[0]))
    print >> context.F, "%s,%d,%.2f,%.2f" % (ts, row[1], row[2], row[3])
context.F.close()
```

RDB に対する SQL 文では標準の関数（substr や count のような）以外に以下の関数を使用することができます。標準の関数以外の関数名はすべてアンダースコアが先頭に付きます。

_year(t)
_month(t)
_day(t)
_hour(t)
_minute(t)
_weekno(t)

引数 t にパフォーマンスデータの日時を表す値（"__TIME"のような）を指定するとそれぞれその日時の年、月、日、時、分、曜日を示す整数を返します（曜日は 0 を日曜日として数えます）。

_strftime(fmt, t)

引数 t に指定したパフォーマンスデータの日時を表す値 t を fmt で指定された文字列で書式化した文字列を返します。fmt 中には以下の書式指定文字が使用可能です。

%d 10 進数で表す月の日付（01～31）
%H 24 時間表記の時間（00～23）
%I 12 時間表記の時間（01～12）
%j 10 進数で表す年初からの日数（001～366）
%m 10 進数で表す月（01～12）
%M 10 進数で表す分（00～59）
%p 実行環境における午前/午後を表す文字列
%U 10 進数で表す週の通し番号。日曜日を週の最初の日とする（00～53）
%w 10 進数で表す曜日（0～6、日曜日が 0）
%W 10 進数で表す週の通し番号。月曜日を週の最初の日とする（00～53）
%y 10 進数で表す西暦の下 2 桁（00～99）
%Y 10 進数で表す 4 桁の西暦
%% パーセント記号

```
_strftime('%Y/%m/%d %H : %M', __TIME)
```

_fmt(fmt, arg0, arg1, ..., argN)

引数 fmt に指定された文字列に従って後続の引数群（可変個指定可能です）を書式化した文字列を返します。fmt 中に '\$@' を記述するとその部分が引数の文字列表現に置換されます。置換は最も左側の '\$@' が arg0、2 番目の '\$@' が arg1 といった順番で行われます。また、fmt 中の '\$\$' は '\$' に置換されます。

```
_fmt('user : $@/command : $@', _USRNAME, _CMDNAME)
```

`_plus(arg0, arg1, ..., argN)`

可変個指定可能な引数 `argX` の値の合計値を返します。ただし、`argX` が 0 未満の値であった場合はその値を合計に含めません。すべての `argX` が 0 未満であった場合は -1 を返します。

`_minus(p, q0, ..., qN)`

`p` が 0 未満であれば -1 を、そうでなければ `p` から以降の引数 (`qX`) を引いた値を返します。ただし、`qX` が 0 未満の場合は計算の対象外とします。

`_multi(arg0, arg1, ..., argN)`

引数すべてを掛けた値を返します。ただし引数中に 0 未満の値があれば -1 を返します。

`_div(num, denom)`

`num` を `denom` で割った値を返します。ただし `num` が 0 未満か `denom` が 0 未満の時は -1 を、`denom` が 0 の時は 0 を返します。

`_avg(p)`

`_sum(p)`

`_min(p)`

`_max(p)`

`_cnt(p)`

引数 `p` で指定された値のそれぞれ平均、合計、最小、最大、件数を返します。ただし 0 未満の値についてはそれを無視して計算を行います。

`_ratio(numer, denom, over)`

`denom` における、`numer` の割合を百分率で返します。ただし `numer` が 0 未満か `denom` が 0 以下の場合は -1 を返します。`over` は割合の閾値です。割合が `over` に指定された値を超えた場合は -1 を返します。`over` が負の値の場合は、閾値の確認を行いません。

`_ratio_rev(numer, denom)`

`denom` における、`denom` と `numer` の差分の割合を百分率で返します。ただし `numer` が 0 未満か `denom` が 0 以下の場合は -1 を返します。また、`numer` が `denom` より大きい場合も -1 を返します。

`_nvl(expr)`

`expr` が数値で 0 以上の場合、`expr` を返します。

上記以外の場合、0 を返します。

`_ptl(expr, ptlpos)`

`expr` のパーセンタイル値を返します。

`ptlpos` にはパーセンタイル位置を示す 0-100 の整数を指定します。

`_corr_abs(expr1, expr2)`

`expr1` と `expr2` の相関係数を絶対値(0.0~1.0)で返します。

サンプリング数が 3 未満の場合、-1 を返します。

`_corr(expr1, expr2)`

`expr1` と `expr2` の相関係数(-1.0~1.0)を返します。

サンプリング数が 3 未満の場合、None を返します。

`_maxof(expr1, expr2)`

`expr1` が最大となるインターバルの `expr2` を返します。
`expr1` に有効値(0 以上)が存在しない場合、-1 を返します。

`_minof(expr1, expr2)`

`expr1` が最小となるインターバルの `expr2` を返します。
`expr1` に有効値(0 以上)が存在しない場合、-1 を返します。

2.3.2. db オブジェクトの属性

以下に db オブジェクトが持つ属性について説明します。これらの属性は `main` 関数の実行時にのみ有効であり、`main_init`、`main_term` 関数の実行時にアクセスした場合の結果は未定義です。

`site`

データ型 : unicode 文字列
値 : 処理対象システムのサイト名

`sys`

データ型 : unicode 文字列
値 : 処理対象システムのシステム名

`osflg`

データ型 : 整数

OS	値
Unix 系	9
Windows 系	10
z/VM	11
i5	12
VMware	13
Hyper-V	14
Virtage	15
上記以外	0

`osname`

データ型 : 文字列
値 : 処理対象システムの OS 名とバージョン

`targetname`

データ型 : 文字列
値 : 処理対象システムの Control Center のターゲット名。Control Center 未使用の場合、値は未定義

`cpunum`

注意 !
この属性は非推奨です。ATCPU.CPUNUM を使用してください。

evalname

データ型 : 文字列

値 : 評価条件ファイル名

stime

データ型 : 整数

値 : 評価対象期間の開始日時を示す 1970/01/01 00:00 からの経過秒数

etime

データ型 : 整数

値 : 評価対象期間の終了日時を示す 1970/01/01 00:00 からの経過秒数

fdate

データ型 : 整数

値 : 評価対象期間の開始日付を示す 8 桁の整数(yyyymmdd)

tdate

データ型 : 整数

値 : 評価対象期間の終了日付を示す 8 桁の整数(yyyymmdd)

fhhmm

データ型 : 整数

値 : 評価対象期間の開始時刻を示す 4 桁以下の整数(HHMM)

thhmm

データ型 : 整数

値 : 評価対象期間の終了時刻を示す 4 桁以下の整数(HHMM)

```
import time, codecs

def main (context, db, param):
    F = codecs.open("C:¥¥foo.txt","w","mbcs")
    print >> F, db.site
    print >> F, db.sys
    print >> F, db.osflg
    print >> F, db.osname
    print >> F, db.targetname
    print >> F, db.cpunum
    print >> F, time.strftime("start=%Y/%m/%d %H:%M", time.localtime(db.stime))
    print >> F, time.strftime("end=%Y/%m/%d %H:%M", time.localtime(db.etime))
    print >> F, "fdate=%08d" % db.fdate
    print >> F, "fhhmm=%04d" % db.fhhmm
    print >> F, "tdate=%08d" % db.tdate
    print >> F, "thhmm=%04d" % db.thhmm
    F.close()
```

2.3.3. db オブジェクトのメソッド

以下に db オブジェクトが持つメソッドとメソッドの呼び出しの結果得られるオブジェクト等について説明します。これらの属性は main 関数の実行時にのみ有効であり、main_init、main_term 関数の実行時にアクセスした場合の結果は未定義です。

getreccount(rec)

rec : (省略不可/レコードオブジェクト)
rec で指定されたレコードの件数を返します。

```
if db.getreccount(ATCPU) == 0:  
    return True
```

getRecordIter(rec)

rec : (省略不可/レコードオブジェクト)
処理対象システムの rec で指定されたレコードへの時系列順のアクセスを提供するオブジェクト (RecordIter オブジェクト) を返します。

RecordIter オブジェクトに for を用いてアクセスすることによりレコードを表現するオブジェクト (SimpleRecord オブジェクト) を順次得ることができます。SimpleRecord オブジェクトには time、intvl という 2 つの属性がありそれぞれレコードの日時とインターバル長を保持しています(*9)。SimpleRecord オブジェクト r にフィールドオブジェクト F を用いて r[F] のように記述とすることで該当フィールドの値を取得することが可能です。

メモ!

(*9) 一部例外があります。例外については後述の「2.8. レコード/フィールドオブジェクトの例外規定」を参照してください。

```
for r in db.getRecordIter(ATCPU):  
    # 日本の日付形式「2007/08/01 20:00」で取得する  
    ymdhm = time.strftime("%Y/%m/%d %H:%M", time.localtime(r.time))  
    intvl = r.intvl  
    usruse = r[ATCPU.USRUSE]
```

getAggregateRow(values, record=None, useptl=True, where="", bases=())

レコードの集約操作結果にアクセスするためのオブジェクト (AggregateRow オブジェクト) を生成して返します。

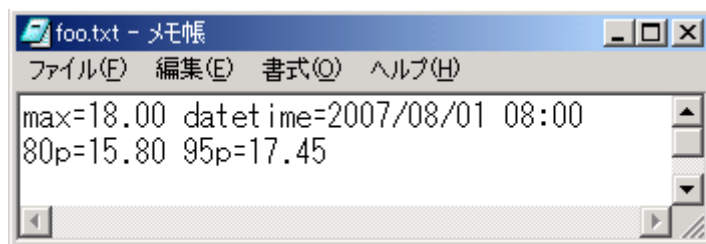
values : (省略不可/文字列型 or フィールドオブジェクト)
集約対象とするパフォーマンスデータ項目をフィールドオブジェクトか RDB の SQL 文の式に相当する文字列で指定します。集約対象とするパフォーマンスデータ項目が複数存在しそれが同一の表を参照する場合はそれらをタプルかリストにまとめることにより一度に集約操作の指定を行うことが可能です。従って values には以下の様な形式の指定が可能です。

```
values=ATCPU.SYSUSE  
values="_USRUSE"  
values=(ATCPU.SYSUSE, "_USRUSE")
```

- record** : (デフォルト : None／レコードオブジェクト)
集約対象のパフォーマンスデータ項目を含むレコードオブジェクトか RDB のテーブルを示す文字列を指定してください。ただし、集約項目 (values 引数) をフィールドオブジェクトで指定した場合はこの引数を省略することが可能です (フィールドオブジェクトが属するレコードオブジェクトが使用されます)。
- useptl** : (デフォルト : True／論理型)
パーセントail値を求める場合には True、求める必要がない場合には False を指定してください。
- where** : (デフォルト : ""／文字列型)
集約対象データの絞り込み条件となる SQL の条件式に相当する文字列を指定してください。
- bases** : (デフォルト : ()／タプル型のリスト)
レコードオブジェクトかレコードオブジェクトのタプル (またはリスト) を指定してください。bases を指定すると bases に指定されたレコードが存在し集約対象のレコードが存在しない場合に集約項目の値を 0 として集約操作を行います。

集約対象となるレコードが存在しない場合、このメソッドは None を返します。

```
context.F = open("C:¥¥foo.txt", "w")
row = db.getAggregateRow("_plus(_USRUSE, _SYSUSE)", record=ATCPU)
if row != None:
    res = row.getResult()
    maxts = time.strftime("%Y/%m/%d %H:%M", time.localtime(res.maxtime))
    print >> context.F, "max=%.2f datetime=%s" % (res.max, maxts)
    print >> context.F, "80p=%.2f 95p=%.2f" % (row.getptl(0.8), row.getptl(0.95))
context.F.close()
```



getAggregateWithKey(record, itemkey, values, useptl=True, where="", bases=(), keyfmt="")

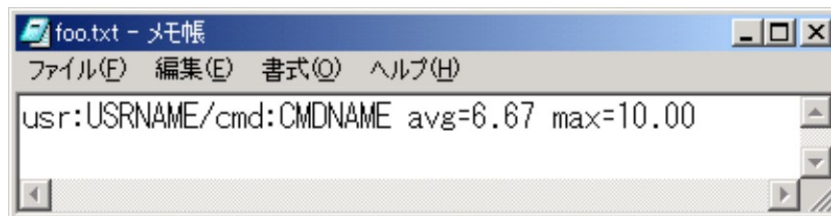
データをキー項目毎に集約した結果を生成するためのオブジェクト (AggregateWithKey オブジェクト) を返します。AggregateWithKey オブジェクトに for を用いてアクセスすることによりキー項目単位の集約結果にアクセスするためのオブジェクト (AggregateRowWithKey オブジェクト) を順次得ることができます。

- record** : (デフォルト : None／レコードオブジェクト)
集約対象のデータ項目を含むレコードオブジェクトか RDB のテーブルを示す文字列を指定してください。
- itemkey** : (デフォルト : 省略不可／文字列型 or レコードオブジェクト)
record で指定した表の中でキーとなる項目を示すフィールドオブジェクトか RDB の SQL 文の式に相当する文字列を指定してください。キーとなる項目が複数ある場合はそれらをタプルかリストにまとめて指定してください。

- values : (省略不可／文字列型 or フィールドオブジェクト)
getAggregateRow メソッドと同様です。
- useptl : (デフォルト : True／論理型)
getAggregateRow メソッドと同様です。
- where : (デフォルト : ""／文字列型)
getAggregateRow メソッドと同様です。
- bases : (デフォルト : ()／タプル型のリスト)
getAggregateRow メソッドと同様です。
- keyfmt : (デフォルト : ""／文字列型)
itemkey で指定した項目の値を書式化するための文字列を指定します。keyfmt 中に '\$@' を記述するとその部分が itemkey で指定した項目の文字列表現に置換されます。itemkey に複数の項目を指定した場合、置換は最も左側の '\$@' が itemkey[0] で置換され、次の '\$@' が itemkey[1] といった順番で行われます。keyfmt 引数に空文字列 ("") を指定した場合は各キー項目をカンマ (,) で区切った形で書式化されます。

```
import codecs

def main (context, db, param):
    context.F = codecs.open("C:¥¥foo.txt","w","mbcs")
    agg = db.getAggregateWithKey(record=ATACCD, itemkey=(ATACCD.USERNAME,
ATACCD.CMDNAME),
                                values=ATACCD.CPUUSE, bases=ATACPU,
keyfmt="usr:$@/cmd:$@")
    for row in agg:
        res = row.getResult()
        print >> context.F, "%s avg=%.2f max=%.2f" % (row.getKey(), res.avg, res.max)
    context.F.close()
```



getAggregateWithKeyBySql(sql, useptl=True, bases=())

データをキー項目毎に集約した結果を生成するためのオブジェクト(AggregateWithKeyBySql オブジェクト)を返します。AggregateWithKeyBySql オブジェクトに for を用いてアクセスすることによりキー項目単位の集約結果にアクセスするためのオブジェクト(AggregateRowWithKey オブジェクト)を順次得ることができます。

sql : (文字列/省略不可)
集約に使用するデータを抽出する為の SQL ステートメントを指定します。第 1 カラムには日時を示す式、第 2 カラムにはキーとなる項目を示す式、第 3 カラム以降には集約対象となる式を指定します。
抽出結果はキー項目と日時で一意になるように、また、抽出順はキー項目、日時の順になるように SQL を記述してください。

AggregateRowWithKey オブジェクトの getResult(..), getptl(..) メソッドの fldidx 引数には第 3 カラムからのオフセットを指定してください(第 3 カラムを 0 として数えます)

useptl と bases 引数の意味は getAggregateWithKey と同じです。

```
sql = ("select t.__TIME, t._NAME, sum(u._USAGEMHZ) "
      "from _VMW_DATACENTER t "
      "inner join _VMW_HOSTCPU u on t.__TIME = u.__TIME "
      "group by t._NAME, t.__TIME "
      "order by t._NAME, t.__TIME")
for agg in db.getAggregateWithKeyBySql(sql):
```

AggregateRow/AggregateRowWithKey オブジェクト

これらのオブジェクトは以下のメソッドを持ちます。

getResult(fldidx=0)
集約結果の平均値や最大値を含むオブジェクト (AggregateResult オブジェクト) を返します。

fldidx : (デフォルト : 0 / 整数型)
getAggregateRow/getAggregateWithKey メソッドの values 引数で指定した集約項目を示すインデックス (values で指定した最も左側の項目を 0 と数えます) を指定します。

getptl(pos, fldidx=0)
pos : (省略不可 / 浮動小数点型)
パーセンタイル位置を示す 0 以上 1 以下の float 型の値を指定します。集約対象項目のパーセンタイル値を返します。

fldidx : (デフォルト : 0 / 整数型)
getResult メソッドと同様です。

getkey()
[AggregateRowWithKey オブジェクトのみ]
getAggregateWithKey メソッドの itemkey 引数で指定した項目を keyfmt 引数により書式化した文字列を返します。

AggregateResult オブジェクト

このオブジェクトは以下の属性を持ちます。

key	集約のキー項目の文字列（getAggregateRow を使用した場合は None）
reccnt	集約項目のレコード件数
basecnt	集約項目のレコードが存在せず bases に指定したレコードが存在した件数
validcnt	0 以上の値の件数
invalidcnt	0 未満の値の件数
min	（0 以上の値の中での）最小値
mintime	上の最小値を記録した日時
max	（0 以上の値の中での）最大値
maxtime	上の最大値を記録した日時
sum	（0 以上の値の）合計値
avg	（0 以上の値の）平均値

loaddata(startdate, enddate, record=[])

db オブジェクトにパフォーマンスデータを追加します。

注意！

この手続きは非推奨です。script_db オブジェクトの loaddata 関数を使用してください。

2.4. script_db オブジェクト

script_db オブジェクトは db オブジェクトの属性、メソッドを継承した拡張モジュール専用のデータベースです。拡張モジュールの main 関数実行毎にデータベースを初期化するため、他モジュールへの影響を考慮せず内容を変更することができます。

script_db オブジェクトは、main 関数内のみ有効なオブジェクトです。main_init/main_term など他関数内の動作は未定義です。

2.4.1. script_db オブジェクトの取得

es1_get_script_db()

script_db オブジェクトを返します。

```
def main(context, db, param):  
    scrip_db = es1_get_script_db()
```

2.4.2. script_db オブジェクトの属性

db オブジェクトと同等です。「2.3. db オブジェクト」を参照してください。

2.4.3. script_db オブジェクトのメソッド

db オブジェクトのメソッドに加え、以下のメソッドが使用可能です。db オブジェクトのメソッドは「2.3. db オブジェクト」を参照してください。

loaddata(startdate, enddate, record=[])

script_db オブジェクトにパフォーマンスデータを追加します。

- startdate : (省略不可／文字列 or int)
追加するパフォーマンスデータの開始時間を"YYYY/MM/DD HH:MM"形式の文字列か、基準時(1970/01/01 00:00:00)からの経過秒数を示す整数値で指定します。
- enddate : (省略不可／文字列 or int)
追加するパフォーマンスデータの終了時間を"YYYY/MM/DD HH:MM"形式の文字列か、基準時(1970/01/01 00:00:00)からの経過秒数を示す整数値で指定します。
- record : (デフォルト: []／リスト)
追加対象とするレコードオブジェクトをリストで指定します。省略した場合は、proc_loadsym に指定したレコードオブジェクトが対象となります。

メモ！

月毎レコードを対象とした場合、startdate/enddate に指定した月のレコードが読み込まれます。

例えば、startdate に 2011/01/01 00:00、enddate に 2011/02/01 00:00 を指定した場合、2011 年 1 月と 2011 年 2 月の月毎レコードが読み込まれます。

```
def main(context, db, param):  
    script_db = es1_get_script_db()  
    startdate = "2010/01/01 00:00"  
    enddate = "2010/01/31 23:59"  
    script_db.loaddata(startdate, enddate, record=[ATCPU])
```

swap_db(memory_ratio=0)

メモリに展開された script_db オブジェクトをディスクにスワップします。

通常 script_db オブジェクトはメモリに配置されています。パフォーマンスデータを追加する際、メモリの使用状況を確認し必要に応じてスワップ処理が自動的に実行されます。この動作は大規模データの追加によるメモリ不足を回避するためです。スワップ処理が実行されない場合、script_db オブジェクトはメモリに残り続け、拡張モジュール自身の使用可能なメモリ領域が制限されます。明示的にスワップ処理を実行することでメモリ領域を確保することができます。

memory_ratio : (省略可 / int or float)

スワップ処理を実行する/しないを判定するための閾値です。物理メモリの空き容量(A)と拡張モジュール自身のメモリ使用量(B)の割合を指定します。メソッド実行時点の割合(C)が指定した数値(A/B)を下回った場合、スワップ処理を実行します。省略した場合もスワップ処理を実行します。

2.5. es1lib モジュール

es1lib モジュールは CS-ADVISOR の評価結果を作成するための機能やレコード（フィールド）オブジェクトを提供します。このモジュールを使用するには拡張モジュール中に

```
from es1lib import *
```

というインポート宣言を記述します。es1lib モジュールは上記の形式のインポート宣言をサポートするようにデザインされています。上記の宣言により拡張モジュール中でレコード（フィールド）オブジェクトの使用が可能になります。レコード（フィールド）オブジェクト以外の es1lib が公開している属性の名はすべて"es1_"または"ES1_"というプレフィックスで始まっています。名前の衝突を避けるために拡張モジュールではすべて英大文字のオブジェクト名（レコードオブジェクトと一致の可能性）や"es1_"または"ES1_"で始まるオブジェクト名は使用しないようにしてください。

以下に es1lib が提供するレコード（フィールド）オブジェクト以外の機能を機能別に示します。

2.5.1. グループの作成

es1_get_msg_resource(resname, filename, bottom=False)

この手続きはメッセージを分類するためのグループをあらわすオブジェクト（MsgResource オブジェクト）を返します。

- resname : (省略不可／文字列型)
グループの名称を表す文字列を指定してください。
- filename : (省略不可／文字列型)
このグループに加えられた内容を保存するためのファイル名(拡張子を除く)を示す文字列を指定してください。実際に作成されるファイル名はここで指定されたファイル名に拡張子（".txt"）を付けたものになります。ただし、指定されたファイル名が"x_"で始まっていない場合は先頭に"x_"を付加します。
- bottom : (デフォルト：False／論理型)
このグループの Performance Web Service 等での表示位置を CS 標準提供処理が出力するグループ群より下にするならば bottom に True を指定してください（bottom=False の場合上に表示されます）。

resname が CS 標準提供処理が出力するグループ名と同一の場合は、このグループに加えられた内容が CS 標準提供処理の出力グループ中にマージされて出力されます。

重要度メッセージが作成（以降「2.5.2. 重要度メッセージの作成」を参照）されなかった MsgResource オブジェクトについては当該グループに「問題ありません。」のメッセージ（Performance Web Service での判定表示は○）が出力されます。

es1_get_log_resource(resname, filename, bottom=False)

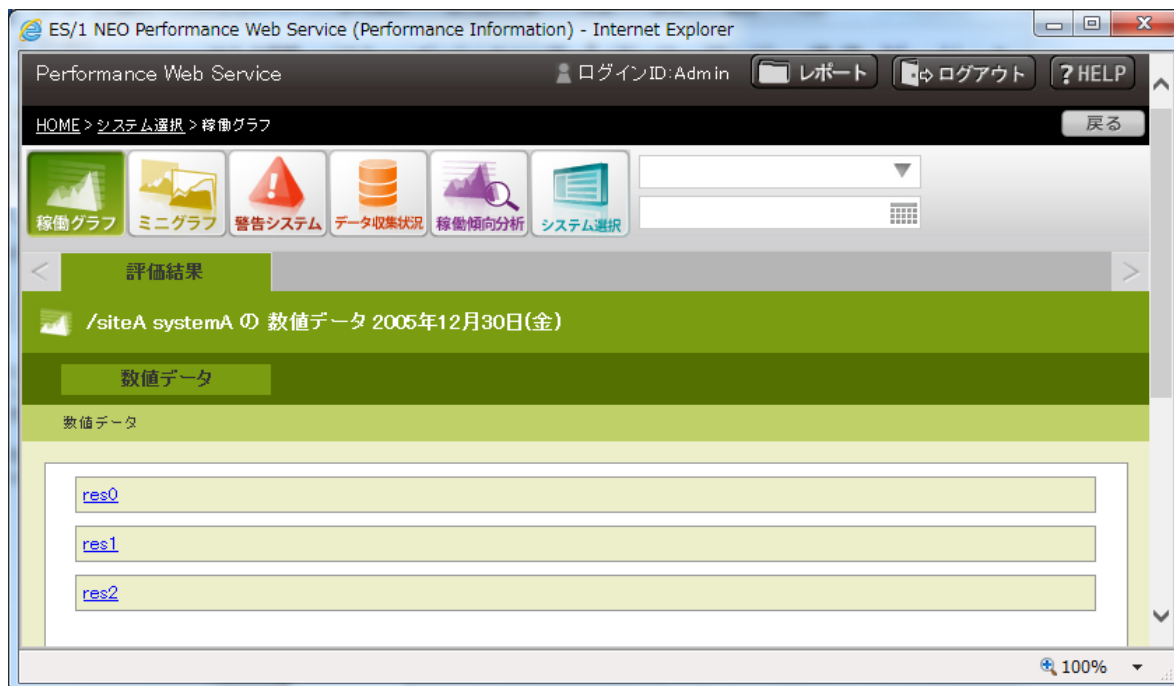
この手続きは数値データを分類するためのグループをあらわすオブジェクト（LogResource オブジェクト）を返します。

resname : (省略不可／文字列型)
es1_get_msg_resource と同様です。

filename : (省略不可／文字列型)
es1_get_msg_resource と同様です。

bottom : (デフォルト：False／論理型)
es1_get_msg_resource と同様です。

```
logres = es1_get_log_resource("res0", "x_res0")
...(中略)
logres = es1_get_log_resource("res1", "x_res1")
...(中略)
logres = es1_get_log_resource("res2", "x_res2")
...(中略)
```



2.5.2. 重要度メッセージの作成

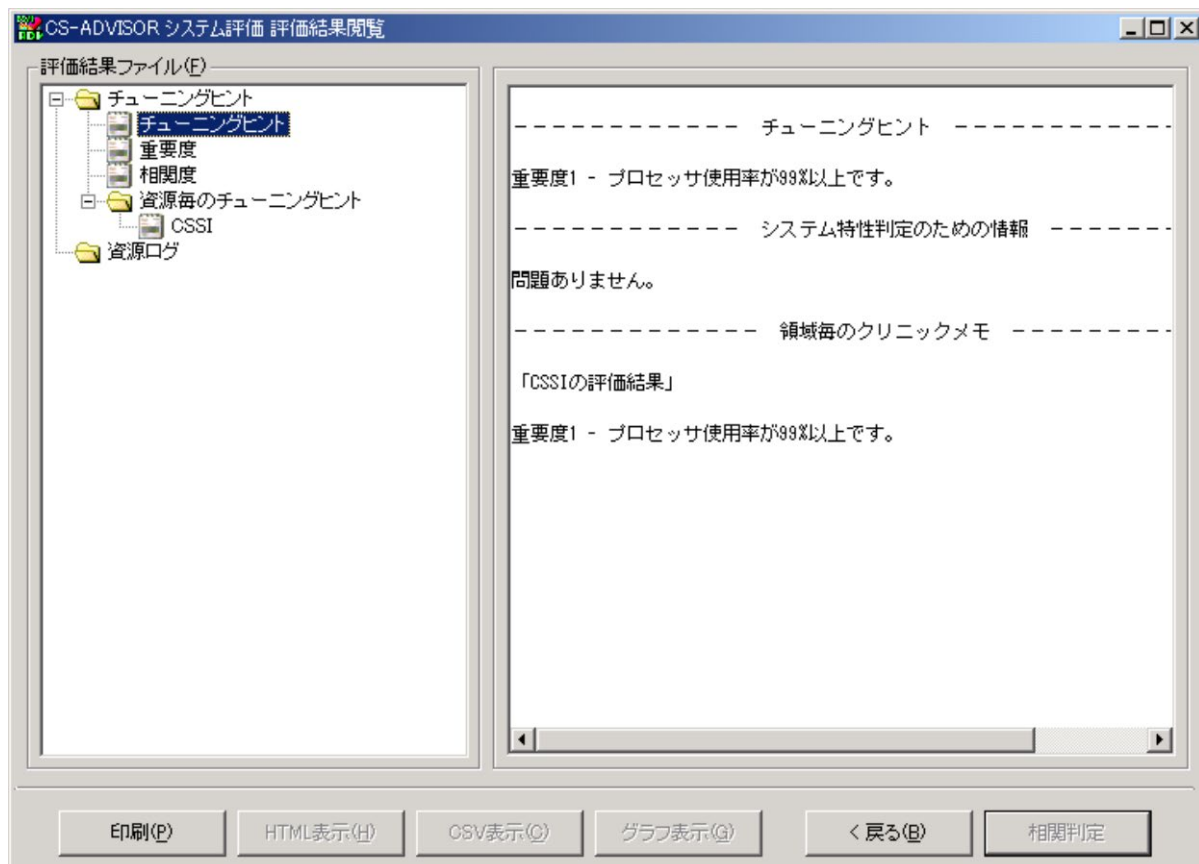
es1_get_msg_resource の呼び出しによって返される MsgResource オブジェクトでは重要度メッセージを作成するための以下のメソッドが使用可能です。

addTuningHintMsg(level, msg, bottom=False)

msg 引数に指定された文字列の重要度メッセージを作成します。

- level : (省略不可／整数型)
以下の 3 つのメソッドの第 1 引数である level には重要度を示す 1 以上 5 以下の整数を指定してください（1 が最も重要とし最上部に表示されます）。
- msg : (省略不可／文字列型)
重要度メッセージを指定してください。文字列¥n を含めると改行を行います。
- bottom : (デフォルト：False／論理型)
最終引数の bottom は MsgResource オブジェクトのグループ名が CS 標準提供処理が出力するグループ名と同一であった場合のメッセージ出力位置の指定であり、True ならば CS 標準提供処理が出力するメッセージより下に、False ならば上に出します。

```
msgres = es1_get_msg_resource('CSSI', filename='x_sample2')  
msgres.addTuningHintMsg(1, u'プロセッサ使用率が 99%以上です。')
```



regTuningHintWithValue(level, msg, value_name="", value_fmt="(\$#)", sort_dsc=True, bottom=False)

重要度メッセージの雛形となるオブジェクト（TuningHintWithValue オブジェクト）を返します。同一の書式で値に相当する部分の文字列が異なる重要度メッセージを複数作成したい場合に使用します。TuningHintWithValue オブジェクトの add メソッドを使用することにより実際に重要度メッセージが作成されます。add メソッドは値に相当する文字列 1 つを引数に受け取ります。

- level** : (省略不可／整数型)
addTuningHintMsg メソッドにおける意味と同じです。
- msg** : (省略不可／文字列型)
重要度メッセージの書式を示す文字列を指定します。文字列¥n を含めると改行を行います。msg 中には'\$#'と'\$#'という書式指定文字が使用可能です。作成したメッセージが Performance Web Service で表示される際には'\$#'の部分が取り除かれて表示され、テキスト形式の評価結果出力時には'\$#'の部分が値を value_fmt で書式化した文字列で置換されて出力されます。'\$#'はいずれの場合も'\$#'に置換されて表示/出力されます。
- value_name** : (デフォルト：""／文字列型)
Performance Web Service の詳細表示画面での列見出しを指定する文字列を指定してください。
- value_fmt** : (デフォルト：(\$#)／文字列型)
値に相当する文字列の書式を指定する文字列です。この文字列中では'\$#'と'\$#'という書式指定文字が使用可能であり、それぞれ、値に相当する文字列と'\$#'に置換されます。
- sort_dsc** : (デフォルト：True／論理型)
作成したメッセージの表示順を値の降順（True）とするか昇順（False）とするかを指定します。
- bottom** : (デフォルト：False／論理型)
addTuningHintMsg メソッドにおける意味と同じです。

```
msgres = es1_get_msg_resource('CSSI test1', filename='x_test_msg1')
hint = msgres.regTuningHintWithValue(1, msg=u"プロセッサ使用率が高いです。$#",
                                     value_name=u"プロセッサ使用率",
                                     value_fmt="( $#% )")
# ソート項目はパディングして桁数を揃えないと予期しない並びになることがあります。
hint.add("%6.2f" % 99.99)
hint.add("%6.2f" % 100.00)
```

上記例の実行の結果は以下ようになります。

(Performance Web Service での表示)

The screenshot shows the Performance Web Service interface in Internet Explorer. The main content area displays tuning hints for 'systemA' on '2012/01/01'. A table lists resources with their status and importance.

資源	判定	重要度: チューニングヒント
CSSI test1	×	1: プロセッサ使用率が高いです。2件 詳細... 100.00, 99.99

Below the table, a detailed view for '重要度1: プロセッサ使用率が高いです。' is shown, including a table for 'プロセッサ 使用率':

プロセッサ 使用率
100.00
99.99

At the bottom, a status message indicates: '重要度1、2のチューニングヒントがあります。' (There are tuning hints for importance 1 and 2).

(CS-ADVISOR での表示)

The screenshot shows the CS-ADVISOR System Evaluation Results window. The left pane shows a tree view of evaluation results, including 'チューニングヒント' (Tuning Hints) and '重要度' (Importance). The right pane displays the details for '重要度1 - プロセッサ使用率が高いです。' (Importance 1 - Processor usage is high).

----- チューニングヒント -----

重要度1 - プロセッサ使用率が高いです。(100.00%)
重要度1 - プロセッサ使用率が高いです。(99.99%)

----- システム特性判定のための情報 -----

問題ありません。

----- 領域毎のクリックメモ -----

「CSSI test1の評価結果」

重要度1 - プロセッサ使用率が高いです。(100.00%)
重要度1 - プロセッサ使用率が高いです。(99.99%)

Buttons at the bottom: 印刷(P), HTML表示(H), CSV表示(C), グラフ表示(G), < 戻る(B), 相関判定

regTuningHintWithItemValue(level, msg, value_name="", item_name="", value_fmt="(\$#)", item_fmt="(\$@)", sort_dsc=True, bottom=False)

重要度メッセージの雛形となるオブジェクト（TuningHintWithItemValue オブジェクト）を返します。同一の書式で値と実体の名前（例えばユーザ名やデバイス名）に相当する部分の文字列が異なる重要度メッセージを複数作成したい場合に使用します。TuningHintWithItemValue オブジェクトの add メソッドを使用することにより実際に重要度メッセージが作成されます。add メソッドは実体の名前に相当する文字列と値に相当する文字列の 2 つを引数に受け取ります。

- level : (省略不可／整数型)
addTuningHintMsg メソッドにおける意味と同じです。
- msg : (省略不可／文字列型)
regTuningHintWithValue メソッドにおける意味と同じです。ただし、'\$#'と'\$ \$'に加えて、'\$@'という書式指定文字が使用可能です。作成したメッセージが Performance Web Service で表示される際には'\$@'の部分が取り除かれて表示され、テキスト形式の評価結果出力時には'\$@'の部分が実体の名前を item_fmt で書式化した文字列で置換されて出力されます。
- value_name : (デフォルト：""／文字列型)
regTuningHintWithValue メソッドにおける意味と同じです。
- item_name : (デフォルト：""／文字列型)
Performance Web Service 詳細表示画面のテーブルの列見出しとなる文字列を指定します。
- value_fmt : (デフォルト："(\$#)"／文字列型)
regTuningHintWithValue メソッドにおける意味と同じです。
- item_fmt : (デフォルト："(\$@)"／文字列型)
実体の名前の書式を指定する文字列です。この文字列中では'\$@'と'\$ \$'という書式指定文字が使用可能であり、それぞれ、実体の名前と'\$'に置換されます。
- sort_dsc : (デフォルト：True／論理型)
regTuningHintWithValue メソッドにおける意味と同じです。
- bottom : (デフォルト：False／論理型)
addTuningHintMsg メソッドにおける意味と同じです。

```
msgres = es1_get_msg_resource('CSSI test1', filename='x_test_msg1')
msg = u'ドライブ$@の使用率が%d%以上です。$# % 90'
hint = msgres.regTuningHintWithItemValue(1, msg, value_name=u'ドライブ使用率(%)',
                                         item_name=u'デバイス名',
                                         value_fmt='( $# % )', item_fmt='( $@ Drive)',
                                         sort_dsc=True, bottom=False)

# ソート項目はパディングして桁数を揃えないと予期しない並びになることがあります。
hint.add("A: FD", "%6.2f" % 99.99)
hint.add("Q: CD", "%6.2f" % 100.00)
```

上記例の実行結果は以下になります。

(Performance Web Service での表示)

The screenshot shows the Performance Web Service interface in a Windows Internet Explorer browser. The page title is "ES/1 NEO Performance Web Service (Performance Information) - Windows Internet Explorer". The user is logged in as "Admin". The main content area displays "評価結果" (Evaluation Results) for "systemA" on "2012/01/01". A table lists tuning hints for resource "CSSI test2", indicating a critical issue (importance 1) related to drive usage. A detailed view shows a table of drive usage percentages.

デバイス名	ドライブ使用率(%)
Q: CD	100.00
A: FD	99.99

At the bottom, a status bar indicates that there are two critical tuning hints (importance 1, 2) and no other issues.

(CS-ADVISOR での表示)

The screenshot shows the CS-ADVISOR System Evaluation Results window. The left pane displays a tree view of evaluation results for "CSSI test1". The right pane shows the detailed tuning hints and system characteristics for "CSSI test1".

チューニングヒント

重要度1 - ドライブ (Q: CD Drive)の使用率が90%以上です。(100.00%)
 重要度1 - ドライブ (A: FD Drive)の使用率が90%以上です。(99.99%)

システム特性判定のための情報

問題ありません。

領域毎のクリニックメモ

「CSSI test1の評価結果」

重要度1 - ドライブ (Q: CD Drive)の使用率が90%以上です。(100.00%)
 重要度1 - ドライブ (A: FD Drive)の使用率が90%以上です。(99.99%)

The bottom of the window contains buttons for "印刷(P)" (Print), "HTML表示(H)" (HTML Display), "CSV表示(C)" (CSV Display), "グラフ表示(G)" (Graph Display), "< 戻る(B)" (Back), and "相関判定" (Correlation Judgment).

regTuningHintWithTable(level, msg, title, titlealign=[], dataalign=[], width=[], bottom=False)

重要度メッセージの雛形となるオブジェクト（TuningHintWithTable オブジェクト）を返します。重要度メッセージに関連する情報をテーブル形式で一覧表示したい場合に使用します。この手続きで一覧表のタイトル行や列幅の指定を行い、TuningHintWithTable オブジェクトの addTableData メソッドを用いて、データ行を追加します。

- p>level : (省略不可／整数型)
-
- addTuningHintMsg メソッドにおける意味と同じです。

p>msg : (省略不可／文字列型)
-
- regTuningHintWithValue メソッドにおける意味と同じです。

p>title : (省略不可／文字列のリスト or タプル)
-
- 一覧表のタイトル行の各要素を指定します。要素数が一覧表の列数になります。列数に制限はありませんが、視認性を考慮し、5 列以下を推奨しています。なお、Performance Web Service に表示される範囲は 5 列までとなります。

p>titlealign : (デフォルト : [ES1_LOG_ALIGN_LEFT,]／ES1_LOG_ALIGN_LEFT、ES1_LOG_ALIGN_RIGHT のリスト or タプル)
-
- 一覧表のタイトル行の各要素の寄せを指定します。省略した場合はすべての列が ES1_LOG_ALIGN_LEFT になります。なお、Performance Web Service に表示される際はここでの指定に関わらず左寄せで表示されます。

ES1_LOG_ALIGN_LEFT : 左寄せ
-
- ES1_LOG_ALIGN_RIGHT : 右寄せ

p>dataalign : (デフォルト : [ES1_LOG_ALIGN_RIGHT,]／ES1_LOG_ALIGN_LEFT、ES1_LOG_ALIGN_RIGHT のリスト or タプル)
-
- 一覧表のデータ行の各要素の寄せを指定します。省略した場合はすべての列が ES1_LOG_ALIGN_RIGHT になります。

p>width : (デフォルト : [0,]／整数のリスト or タプル)
-
- 一覧表の各列の幅を半角文字数(バイト数)で指定します。指定可能な値は 2 以上の正の整数です。省略した場合と 0 を指定した場合、列中の最大の長さを持つ要素を基準にして自動的に幅が決定されます。この指定は CS-ADVISOR の評価結果閲覧画面に表示される評価結果に反映されます。なお、Performance Web Service 上の表示はブラウザの動作に依存するため、明示的に指定することはできません。

p>bottom : (デフォルト : False／論理型)
-
- addTuningHintMsg メソッドにおける意味と同じです。

addTableData(*args)

TuningHintWithTable オブジェクトにデータ行を追加するメソッドです。

***args** : (省略不可／文字列のリスト or タプル)
一覧表のデータ行の文字列を行毎にリストかタプルで指定します。1 行～複数行の指定が可能です。

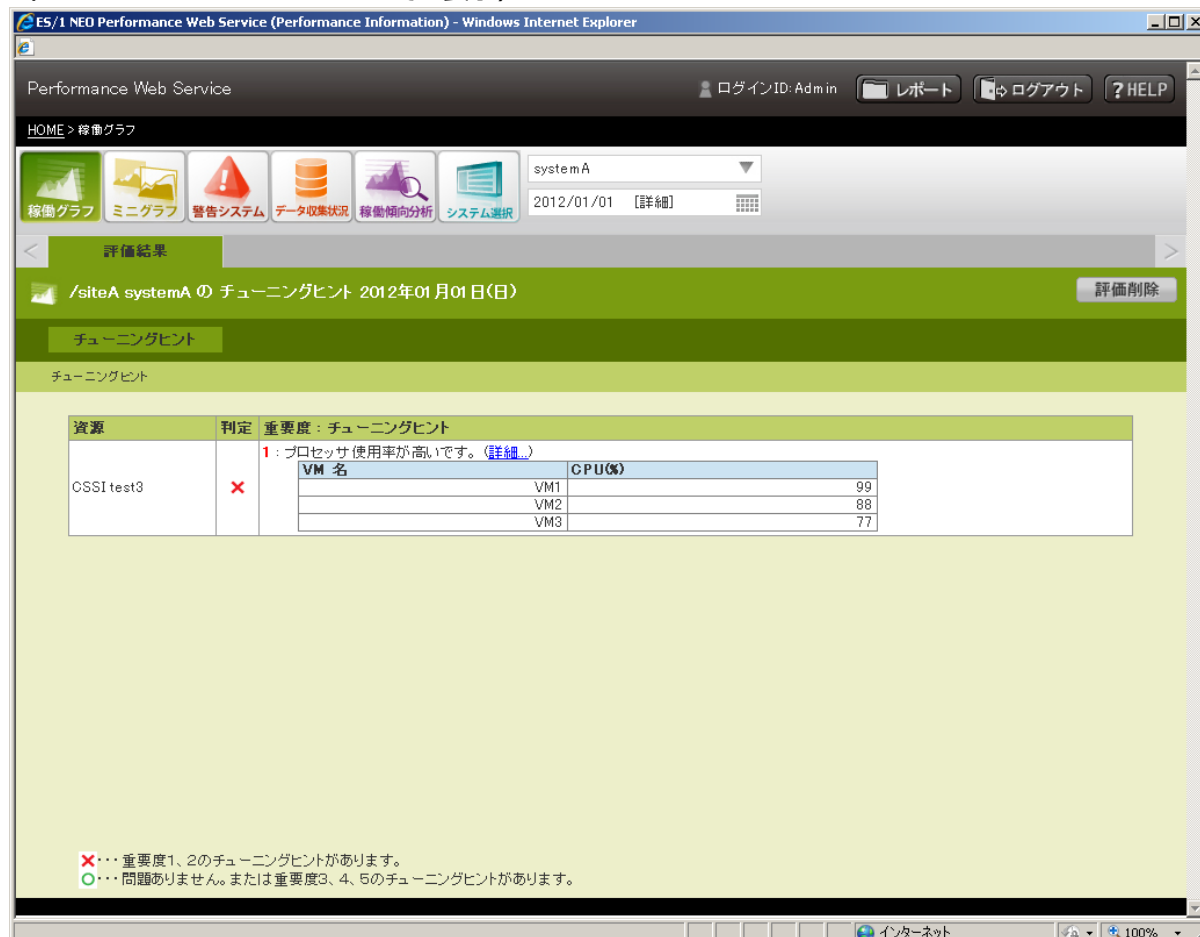
```
msgres = es1_get_msg_resource(u'CSSI test3', filename=u'x_test_msg3')

level = 1
msg = u'プロセッサ使用率が高いです。'
title = [u'VM 名', u'CPU(%)']
width = [10, 10]
hint = msgres.regTuningHintWithTable(level, msg, title, width=width)

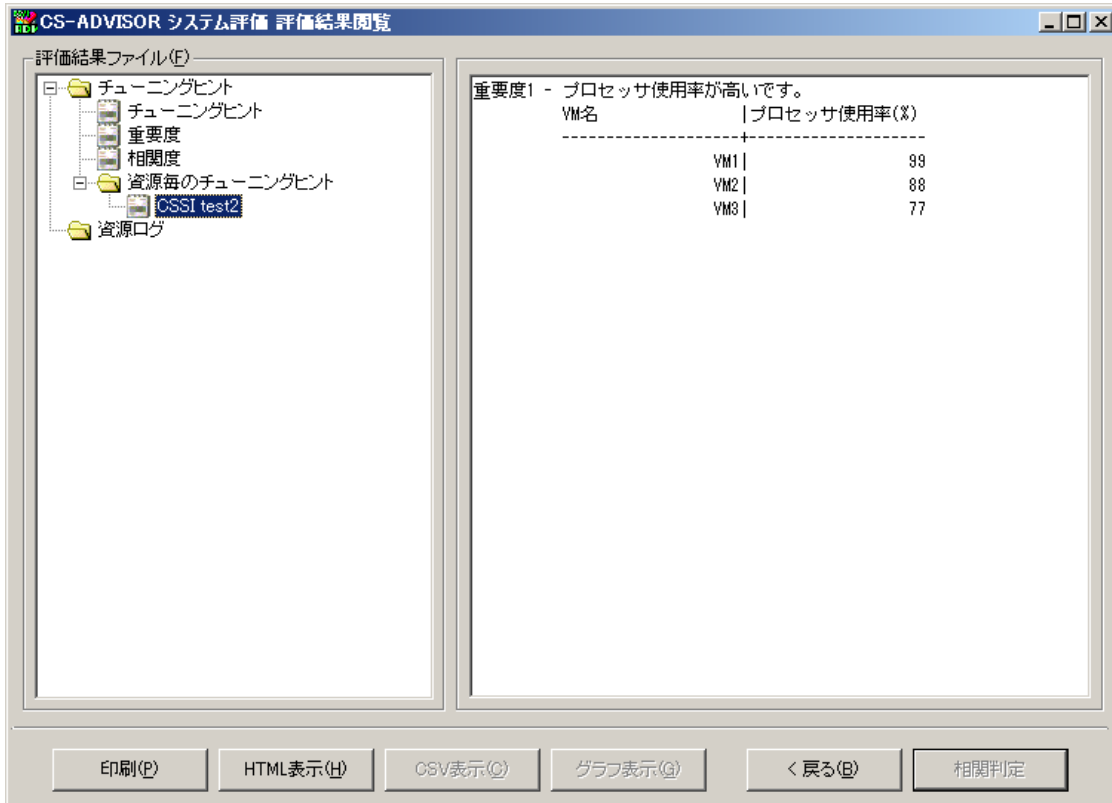
data0 = (u"VM1", u"99")
data1 = (u"VM2", u"88")
data2 = (u"VM3", u"77")
hint.addTableData( data0, data1, data2 )
```

上記例の実行結果は以下になります。

(Performance Web Service での表示)



(CS-ADVISOR での表示)



addSummary(title, titlealign=[], dataalign=[], width=[], msg=None)

TuningHintWithTable オブジェクトにサマリ画面用の重要度メッセージ、一覧表を追加するメソッドです。

メモ！

TuningHintWithTable オブジェクトで作成した重要度メッセージ、テーブルは Performance Web Service のサマリ画面、詳細画面に表示されます。

詳細画面と異なる内容をサマリ画面に表示したい場合に、このメソッドを使用します。

- title** : (省略不可／文字列のリスト or タプル)
regTuningHintWithTable メソッドにおける意味と同じです。
- titlealign** : (デフォルト : [ES1_LOG_ALIGN_LEFT,]／ES1_LOG_ALIGN_LEFT、ES1_LOG_ALIGN_RIGHT のリスト or タプル)
regTuningHintWithTable メソッドにおける意味と同じです。
- dataalign** : (デフォルト : [ES1_LOG_ALIGN_RIGHT,]／ES1_LOG_ALIGN_LEFT、ES1_LOG_ALIGN_RIGHT のリスト or タプル)
regTuningHintWithTable メソッドにおける意味と同じです。
- width** : (デフォルト : [0,]／整数のリスト or タプル)
regTuningHintWithTable メソッドにおける意味と同じです。
- msg** : (デフォルト : TuningHintWithTable.msg／文字列型)
サマリ画面用の重要度メッセージを指定します。省略した場合は詳細画面と同じです。

addTableDataSummary(*args)

addSummary メソッドで作成したサマリ画面用の一覧表にデータ行を追加するメソッドです。

*args : (省略不可／文字列のリスト or タプル)
addTableData メソッドと同じです。

2.5.3. 数値データの作成

es1_get_log_resource の呼び出しによって返される LogResource オブジェクトでは数値データを作成するための以下のメソッドが使用可能です。

addLog(logname="", title=(), titlealign=(), dataalign=(), width=(), topnum=False, bottom=0)

数値データを出力するための表の雛形をあらわすオブジェクト (ResourceLog オブジェクト) を生成して返します。

logname : (省略不可／文字列型)
表のタイトルとして表示される文字列を指定します。

title : (省略不可／タプル型 or リスト型)
タプル (リスト) の各要素には表の見出し行の各列の見出し文字列を指定します。

titlealign : (デフォルト : ES1_LOG_ALIGN_LEFT／タプル型 or リスト型)
タプル (リスト) の各要素には表の見出し行の各列の見出し文字列の整列方法を指定します (*10)。

dataalign : (デフォルト : ES1_LOG_ALIGN_RIGHT／タプル型 or リスト型)
タプル (リスト) の各要素には表の見出し行以外の行における各列の文字列の整列方法を指定します (*10)。

width : (デフォルト : 0／タプル型 or リスト型)
タプル (リスト) の各要素にはテキストの評価結果での各列の幅を半角文字の数で指定します (ここに 0 以下の値を指定すると列中の最大の長さを持つ項目を基準にして自動的に幅が決定されます)。

topnum : (デフォルト : False／論理型)
CS の動作環境設定におけるデバイス・コマンド情報等の表示制限で指定した件数に表の最大行数を制限する場合に True を指定します。

bottom : (デフォルト : False／論理型)
LogResource オブジェクトのグループ名が CS 標準提供処理が出力するグループ名と同一であった場合の表の出力位置の指定であり、True ならば CS 標準提供処理が出力する表より下に、False ならば上に出力します。

メモ !

(*10) 整列方法には ES1_LOG_ALIGN_LEFT (左寄せ)、ES1_LOG_ALIGN_RIGHT (右寄せ)、ES1_LOG_ALIGN_CENTER (中央寄せ) のいずれかを指定してください。また、テキストの評価結果 (資源ログ) においては ES1_LOG_ALIGN_CENTER は右寄せとなります。

実際に出力される表の列の数は ResourceLog オブジェクトの addTableData メソッド（後述）によって作成される表の各行の列数と title に指定されたタプル（リスト）の要素数によって決定されます。いずれかの内で最大の数が実際に出力される表の列数になります。title、titlealign、dataalign、width の要素数が実際に出力される表の列数よりも少ない場合は残りの列についてはデフォルト値が使用されます。title は空文字列（""）、titlealign と dataalign は ES1_ALIGN_RIGHT、width は 0 がそれぞれのデフォルト値となります。また、titlealign、dataalign、width の要素数が実際に出力される表の列数よりも多い場合は、表の列数を超える部分については無視されます。

ResourceLog オブジェクト

このオブジェクトは以下のメソッドを持ちます。

addTableData(row)

表にデータ行 1 行を作成するためのメソッドです。

row : 行の各列の値を示す文字列からなるタプル（リスト）を指定します。

```
logres = es1_get_log_resource("res0", "x_res0")
log=logres.addLog(logname="min/max",title=("item","min","max"),titlealign=(ES1_LOG_ALIGN_LEFT,
',
    ES1_LOG_ALIGN_LEFT,ES1_LOG_ALIGN_LEFT), dataalign=(ES1_LOG_ALIGN_RIGHT,
    ES1_LOG_ALIGN_RIGHT,ES1_LOG_ALIGN_RIGHT), width=(10, 10, 10), topnum=False,
bottom=False)
log.addTableData(("foo", "0", "99"))
log.addTableData(("bar", "10", "100"))
```

上記例の実行結果は以下のようになります。

(Performance Web Service での表示)

The screenshot shows the 'Performance Web Service' interface in a Windows Internet Explorer browser. The page title is 'ES/1 NEO Performance Web Service (Performance Information)'. The breadcrumb navigation is 'HOME > 稼働グラフ'. The main content area is titled '評価結果' (Evaluation Results) and shows data for '/siteA systemA の 数値データ 2012年01月01日(日)'. Below this, there are tabs for 'チューニングヒント' (Tuning Hints) and '数値データ' (Numerical Data). The '数値データ' tab is selected, showing a table for 'res0' with columns 'item', 'min', and 'max'. The table contains two rows: 'foo' with min=0 and max=99, and 'bar' with min=10 and max=100. There are links for '別ウィンドウで表示' (Display in another window) and 'CSVダウンロード' (Download CSV) above the table.

item	min	max
foo	0	99
bar	10	100

(CS-ADVISOR での表示)

The screenshot shows the 'CS-ADVISOR システム評価 評価結果閲覧' (CS-ADVISOR System Evaluation Results Review) window. On the left, there is a file tree under '評価結果ファイル(F)' (Evaluation Results Files) showing a hierarchy: 'チューニングヒント' (Tuning Hints) > 'チューニングヒント' (Tuning Hints) > '重要度' (Importance) > '相関度' (Correlation) > '資源ログ' (Resource Log) > 'res0'. On the right, there is a table with columns 'min/max', 'item', 'min', and 'max'. The table contains two rows: 'foo' with min=0 and max=99, and 'bar' with min=10 and max=100. At the bottom, there are buttons for '印刷(P)' (Print), 'HTML表示(H)' (HTML Display), 'CSV表示(C)' (CSV Display), 'グラフ表示(G)' (Graph Display), '< 戻る(B)' (Back), and '相関判定' (Correlation Judgment).

min/max	item	min	max
	foo	0	99
	bar	10	100

`sortbystr(col, order=ES1_SORT_ASC)`

このメソッドの呼び出し以前に `addTableData` で追加された行をソートします。

col : (省略不可／整数型)
ソートは各行の `col` で指定された番号（最も左側のカラムを 0 と数えます）のカラムの文字列を基に行われます。

order : (デフォルト: `ES1_SORT_ASC`／整数型)
`ES1_SORT_ASC` が指定された場合は昇順で、`ES1_SORT_DSC` が指定された場合は降順でソートを行います。

`sortbynum(col, order=ES1_SORT_DSC)`

このメソッドの呼び出し以前に `addTableData` で追加された行をソートします。

col : (省略不可／整数型)
ソートは各行の `col` で指定された番号(最も左側のカラムを 0 と数えます)のカラムの文字列を数値に変換した値を基に行われます。

order : (デフォルト: `ES1_SORT_DSC`／整数型)
`ES1_SORT_ASC` が指定された場合は昇順で、`ES1_SORT_DSC` が指定された場合は降順でソートを行います。

複数のキーを用いたソートは `sortbystr/sortbynum` を複数回呼び出すことにより実現できます。この場合、優先順位が高いキーによるソートを後に呼び出します。

```
…中略(addLog 関数を参照してください)…  
log.addTableData(("aaa", "1", "99"))  
log.addTableData(("ccc", "2", "100"))  
log.addTableData(("bbb", "3", "50"))  
log.addTableData(("aaa", "4", "99"))  
log.sortbystr(0, ES1_SORT_DSC)  
log.sortbynum(2, ES1_SORT_ASC)
```

上記例の実行結果は以下ようになります。

(Performance Web Service での表示)

The screenshot shows the 'Performance Web Service' interface in a Windows Internet Explorer browser. The page title is 'ES/1 NEO Performance Web Service (Performance Information) - Windows Internet Explorer'. The user is logged in as 'Admin'. The main content area displays '評価結果' (Evaluation Results) for 'system A' on '2012/01/01'. Below this, there are tabs for 'チューニングヒント' (Tuning Hints) and '数値データ' (Numerical Data). The '数値データ' tab is selected, showing a table of evaluation results for 'res1'.

item	min	max
bbb	3	50
aaa	1	99
aaa	4	99
ccc	2	100

(CS-ADVISOR での表示)

The screenshot shows the 'CS-ADVISOR システム評価 評価結果閲覧' (CS-ADVISOR System Evaluation Results Viewer) window. The left pane shows a tree view of evaluation results, including 'チューニングヒント' (Tuning Hints) and '重要度' (Importance). The right pane displays a table of evaluation results for 'res1'.

item	min	max
bbb	3	50
aaa	1	99
aaa	4	99
ccc	2	100

addLogMulti(logname="", title=(), width_layout=ES1_LOG_WIDTH_AUTO, width_columns={}, topnum=False, bottom=False)

数値データを出力するための表の雛形をあらわすオブジェクト（ResourceLogMulti オブジェクト）を生成して返します。
addLog と異なり、見出し項目の段組み(結合)とセルの装飾を行うことができます。

logname : (省略不可／文字列型)
上記の addLog と同様です。

title : (省略不可／タプル型 or リスト型)
タプル (リスト) の各要素に、表の見出し行の各列の見出し文字列、または Cell オブジェクト(後述)を指定します。見出し行を段組みする場合は、(中項目, (小項目, 小項目, …))の形式で指定します。

width_layout : (デフォルト : ES1_LOG_WIDTH_AUTO)
表全体の列幅を指定します。ES1_LOG_WIDTH_AUTO が指定された場合は「自動」、ES1_LOG_WIDTH_FIXED が指定された場合は「等幅」になります。このオプションは PWS 閲覧時のみ有効です。

width_columns : (デフォルト : {} / {<column_index>:<value>})
個別に列幅を指定します。<column_index>は対象とする列のインデックス(0～n 番目)を、<value>は「px」で指定する場合は整数を、「%」で指定する場合は 1 以下の少数を指定します。このオプションは PWS 閲覧時のみ有効です。

3 番目の列を「100px」に : width_columns = {2: 100}
5 番目の列を「20%」に : width_columns = {4: .20}

topnum : (デフォルト : False／論理型)
上記の addLog と同様です。

bottom : (デフォルト : False／論理型)
上記の addLog と同様です。

メモ！

見出し行に 3 段以上の段組みを指定することはできません。

ResourceLogMulti オブジェクト

このオブジェクトは以下のメソッドを持ちます。

addTableData(*rows)

表にデータ行 1 行を作成するためのメソッドです。

rows : (省略不可／タプル型 or リスト型)
行の各列の値を示す文字列、または Cell オブジェクト(後述)をタプル (リスト) で指定します。

addTableDataMulti(head, *rows)

表にデータ行を作成するためのメソッドです。データ行で段組みを行う場合に使用します。

- head : (省略不可／unicode)
段組みの中項目とする文字列、または Cell オブジェクト(後述)を指定します。
- rows : (省略不可／タプル型 or リスト型)
行の各列の値を示す文字列、または Cell オブジェクト(後述)をタプル (リスト) で指定します。
タプルの第 1 要素が小項目になります。

メモ！

addTableData と addTableDataMulti を同時に使用することはできません。

addCsvData(csvdata)

Performance Web Service 閲覧画面からダウンロードする CSV データを作成するためのメソッドです。このメソッドを使用しない場合、自動で CSV データを生成します。

csvdata : (省略不可／unicode)
表を表す文字列を csv 形式で指定します。

es1_cell(value, align=None, valign=None, color=None, bgcolor=None, style=None)

表中の各セルを装飾する Cell オブジェクトを返します。

value : (省略不可／文字列型)
セルの値を指定します。

align : (省略可／ES1_LOG_ALIGN_LEFT or ES1_LOG_ALIGN_CENTER or ES1_LOG_ALIGN_RIGHT)
セルの水平方向の寄せを指定します。省略した場合、以下になります。
見出し行 ES1_LOG_ALIGN_CENTER(中央寄せ)
データ行 ES1_LOG_ALIGN_RIGHT(右寄せ)

valign : (省略可／ES1_LOG_VALIGN_TOP or ES1_LOG_VALIGN_MIDDLE or ES1_LOG_VALIGN_BOTTOM)
セルの垂直方向の寄せを指定します。省略した場合、ES1_LOG_VALIGN_MIDDLE になります。

color : (省略可／文字列型)
セルの文字の色をカラー名か RGB 値で指定します。

bgcolor : (省略可／文字列型)
セルの背景色をカラー名か RGB 値で指定します。

style : (省略可／文字列型)
セルの装飾をスタイルシート形式で指定します。style を指定した場合、align 等の指定は無効になります。

```
logres = es1_get_log_resource(u"multi", u'x_multi')

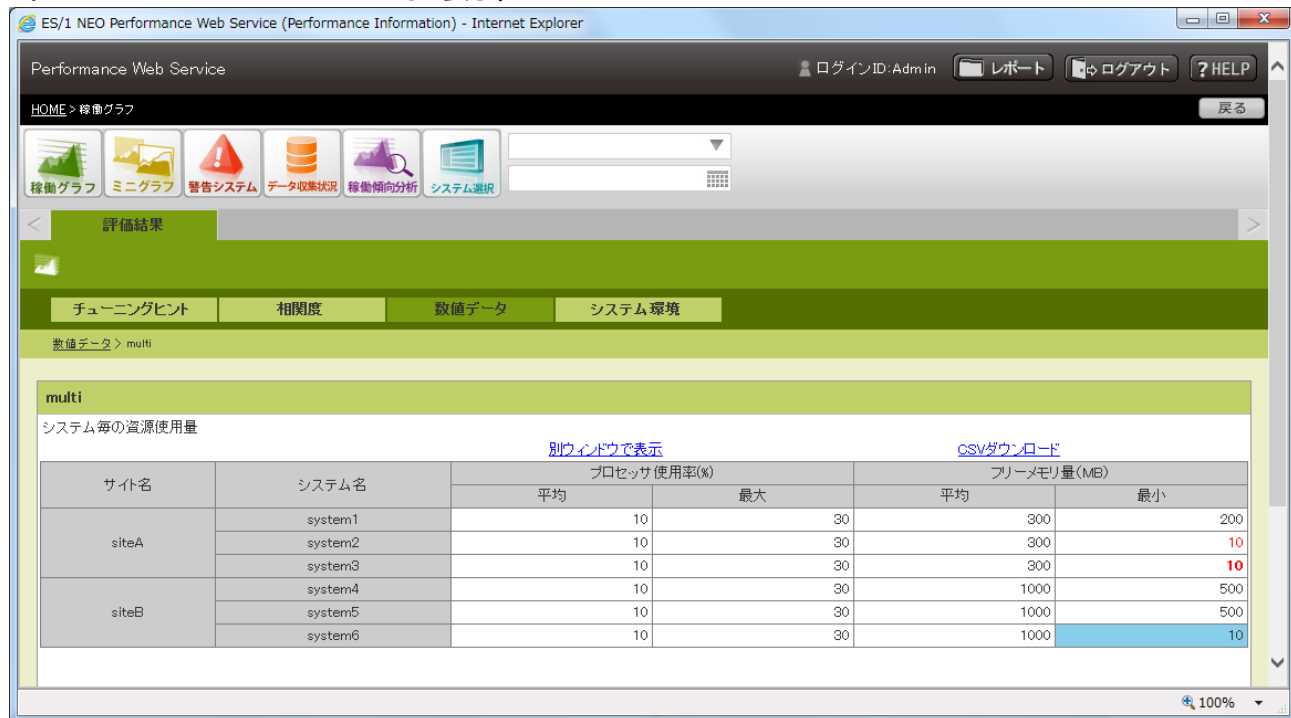
log = logres.addLogMulti(logname=u"システム毎の資源使用量",
                        title=(u"サイト名",
                              u"システム名",
                              (u"プロセッサ使用率(%)", (u"平均", u"最大")),
                              (u"フリーメモリ量 (MB) ", (u"平均", u"最小")),
                              ),
                        )

log.addTableDataMulti( u"siteA",
                      (u"system1", "10", "30", "300", "200"),
                      (u"system2", "10", "30", "300", es1_cell("10", color="red")),
                      (u"system3",      "10",      "30",      "300",      es1_cell("10",
style="color:red;font-weight:bold;")),
                      )

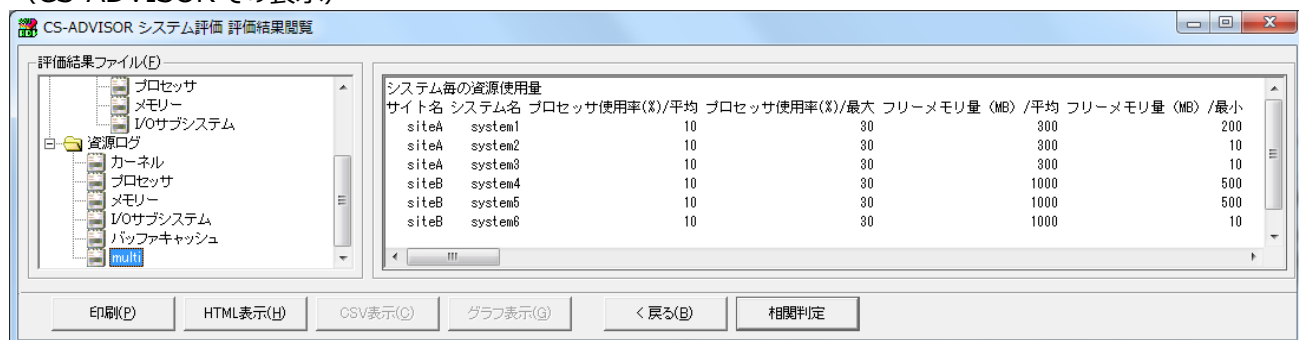
log.addTableDataMulti( u"siteB",
                      (u"system4", "10", "30", "1000", "500"),
                      (u"system5", "10", "30", "1000", "500"),
                      (u"system6", "10", "30", "1000", es1_cell("10", bgcolor="skyblue")),
                      )
```

上記例の実行結果は以下ようになります。

(Performance Web Service での表示)



(CS-ADVISOR での表示)



2.5.4. 相関度メッセージの作成

相関度メッセージを作成するために以下の手続きが使用可能です。

es1_CorrTerm(termtext, value, record=None, where="", bases=(), use_script_db=False)

表のレコード全体を基にした相関判定対象項目をあらわすオブジェクト（CorrTerm オブジェクト）を生成して返します。

- termtext** : (省略不可／文字列型)
相関度メッセージに埋め込まれる相関判定対象項目の名称を指定します。
- value** : (省略不可／フィールドオブジェクト or 文字列型)
相関判定対象項目の値を示すフィールドオブジェクトか RDB の SQL 文の式に相当する文字列を指定してください。
- record** : (デフォルト：None／レコードオブジェクト or 文字列型)
相関判定対象項目を含むレコードオブジェクトか RDB のテーブルを示す文字列を指定してください。ただし、相関判定対象項目の値（value 引数）をフィールドオブジェクトで指定した場合はこの引数を省略することが可能です（フィールドオブジェクトが属するレコードオブジェクトが使用されます）。
- where** : (デフォルト：""／文字列型)
相関判定対象項目のレコードの絞り込み条件となる SQL の条件式に相当する文字列を指定してください。bases にはレコードオブジェクトかレコードオブジェクトのタプル（またはリスト）を指定してください。
- bases** : (デフォルト：()／レコードオブジェクト or レコードオブジェクトのシーケンス)
bases を指定すると相関判定対象項目のレコードが存在せず bases に指定されたレコードが存在したインターバルの値を 0 として相関判定を行います。
- use_script_db** : (デフォルト：False／論理型)
True を指定すると script_db オブジェクトを対象として相関判定を行います。

サンプルソース及びサンプル出力結果は es1_add_corrmsg 関数にまとめて記載いたします。

es1_CorrTermWithItem(termtext, record, itemkey, value, where="", bases=(), keyfmt="", item_name="", item_fmt="(\$@)", use_script_db=False)

表のレコードを特定項目でグルーピングした相関判定対象項目をあらわすオブジェクト（CorrTermWithItem オブジェクト）を生成して返します。

- termtext** : (省略不可／文字列型)
相関度メッセージに埋め込まれる相関判定対象項目の名称を指定します。termtext 中には'\$@'と'\$ \$'という書式指定文字が使用可能です。相関度メッセージが Performance Web Service で表示される際には'\$@'の部分が取り除かれて表示され、テキスト形式の評価結果出力時には'\$@'の部分がグルーピングのキーとなる項目（後述の itemkey）を書式化した文字列で置換されて出力されます。'\$ \$'はいずれの場合も'\$'に置換されて表示/出力されます。
- record** : (省略不可／レコードオブジェクト or 文字列型)
相関判定対象項目を含むレコードオブジェクトか RDB のテーブルを示す文字列を指定してください。

- itemkey** : (省略不可／フィールドオブジェクト or 文字列型)
record で指定した表のレコード中でキーとなる項目を示すフィールドオブジェクトか RDB の SQL 文の式に相当する文字列を指定してください。キーとなる項目が複数ある場合はそれらをタプルかリストにまとめて指定してください。
- value** : (省略不可／フィールドオブジェクト or 文字列型)
上記の es1_CorrTerm と同様です。
- where** : (デフォルト: ""／文字列型)
上記の es1_CorrTerm と同様です。
- bases** : (デフォルト: ()／レコードオブジェクト or レコードオブジェクトのシーケンス)
上記の es1_CorrTerm と同様です。
- keyfmt** : (デフォルト: ""／文字列型)
itemkey で指定した項目の値を書式化するための文字列を指定します。keyfmt 中に '\$@' を記述するとその部分が itemkey で指定した項目の文字列表現に置換されます。itemkey に複数の項目を指定した場合、置換は最も左側の '\$@' が itemkey[0] で置換され、次の '\$@' が itemkey[1] といった順番で行われます。keyfmt 引数に空文字列 ("") を指定した場合は各キー項目をカンマ (,) で区切った形で書式化されます。keyfmt で書式化された文字列は Performance Web Service 詳細表示画面のテーブルにそのまま表示されます。
- item_name** : (デフォルト: ""／文字列型)
Performance Web Service 詳細表示画面のテーブルの列見出しとなる文字列を指定します。
- item_fmt** : (デフォルト: "(\$@)"／文字列型)
keyfmt で書式化された文字列をテキストの相関度メッセージ中に埋め込む時の書式を指定する文字列です。item_fmt 中の '\$@' は keyfmt の結果文字列で置換され、その結果文字列が termtext 中の '\$@' を置換します。
- use_script_db** : (デフォルト: False／論理型)
True を指定すると script_db オブジェクトを対象として相関判定を行います。

サンプルソース及びサンプル出力結果は es1_add_corrmsg 関数にまとめて記載いたします。

es1_CorrTermByList(termtext, values, bases=())

相関判定対象項目をあらわすオブジェクト（CorrTermByList オブジェクト）を生成して返します。db 表中のレコードを対象とする es1_CorrTerm と異なり、任意の値を指定することができます。

- termtext : (省略不可／文字列型)
 上記の es1_CorrTerm と同様です。
- values : (省略不可／リスト)
 相関判定対象項目の「時間」と「値」の 2 次元配列を指定します。[[time, value], ...]
 time 1970/01/01 00:00 からの経過秒数
 value 値
- bases : (デフォルト: ())／レコードオブジェクト or レコードオブジェクトのシーケンス
 上記の es1_CorrTerm と同様です。

サンプルソース及びサンプル出力結果は add_corrmsg 関数にまとめて記載いたします。

es1_CorrTermWithItemByList(termtext, values, bases=(), keyfmt="", item_name="", item_fmt="(\$@)")

キー項目でグルーピングした相関判定対象項目をあらわすオブジェクト（CorrTermWithItemByList オブジェクト）を生成して返します。db 表中のレコードを対象とする es1_CorrTermWithItem と異なり、任意の値を指定することができます。

termtext : (省略不可／文字列型)
上記の es1_CorrTermWithItem と同様です。

values : (省略不可／リスト)
相関判定対象項目の「時間」と「キー」と「値」を 2 次元配列で指定します。[[time, key, value], ...]
time 1970/01/01 00:00 からの経過秒数
key unicode 文字列
value 値

bases : (デフォルト：()／レコードオブジェクト or レコードオブジェクトのシーケンス)
上記の es1_CorrTermWithItem と同様です。

keyfmt : (デフォルト：""／文字列型)
上記の es1_CorrTermWithItem と同様です。

item_name : (デフォルト：""／文字列型)
上記の es1_CorrTermWithItem と同様です。

item_fmt : (デフォルト："(\$@)"／文字列型)
上記の es1_CorrTermWithItemKey と同様です。

サンプルソース及びサンプル出力結果は add_corrmsg 関数にまとめて記載いたします。

es1_add_corrmsg(leftterm, rightterm, msgres=None, logres=None, bottom=False)

相関度メッセージの作成を行います。

leftterm : (省略不可／CorrTerm オブジェクト or CorrTermWithItem オブジェクト)
相関判定に使用する CorrTerm オブジェクトか CorrTermWithItem オブジェクトを指定します
(leftterm と rightterm が同一のオブジェクトの場合はエラーとなります)。

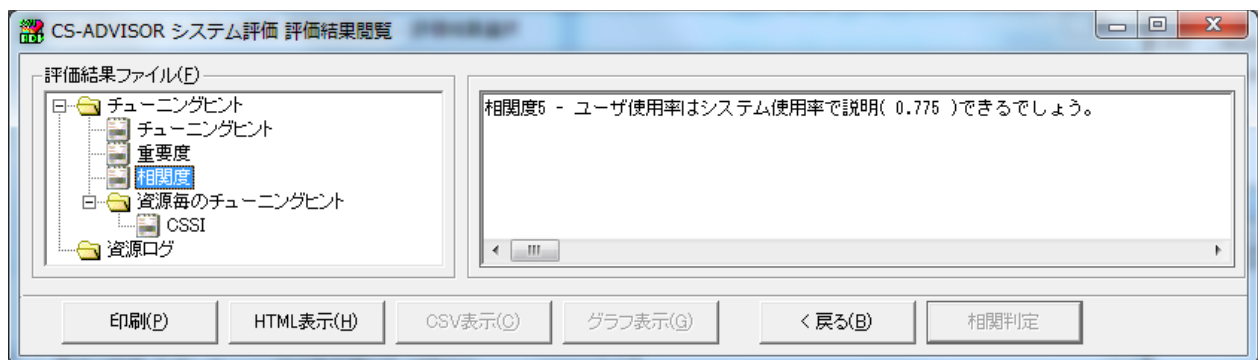
rightterm : (省略不可／CorrTerm オブジェクト or CorrTermWithItem オブジェクト)
相関判定に使用する CorrTerm オブジェクトか CorrTermWithItem オブジェクトを指定します
(leftterm と rightterm が同一のオブジェクトの場合はエラーとなります)。

msgres : (デフォルト：None／MsgResource オブジェクト and LogResource オブジェクト)
その相関度メッセージが出力されるグループを示す MsgResource オブジェクトと LogResource オブジェクトをそれぞれ指定します (LogResource 指定の結果としての相関度メッセージの出力はテキストの評価結果中にのみ行われ、Performance Web Service の評価結果表示には影響しません)。

- logres** : (デフォルト : None / MsgResource オブジェクト and LogResource オブジェクト)
その相関度メッセージが出力されるグループを示す MsgResource オブジェクトと LogResource オブジェクトをそれぞれ指定します (LogResource 指定の結果としての相関度メッセージの出力はテキストの評価結果中にのみ行われ、Performance Web Service の評価結果表示には影響しません)。
- bottom** : (デフォルト : False / 論理型)
LogResource オブジェクトが CS 標準提供処理の出力グループ名と同一であった場合の出力位置の指定であり、True ならば CS 標準提供処理が出力する相関度メッセージより下に、False ならば上に出力します。

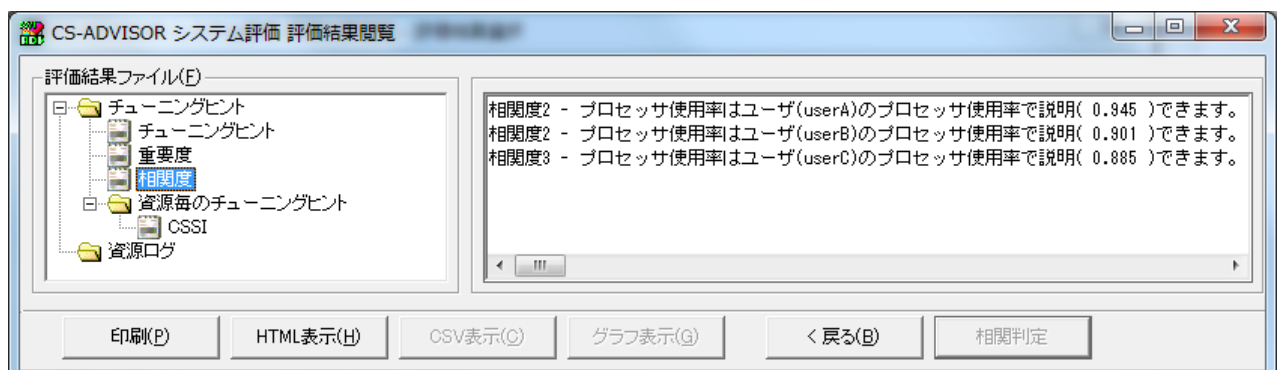
(例 1) es1_CorrTerm を用いた相関判定

```
# ATCPU.USRUSE と ATCPU.SYSUSE で相関判定を行いメッセージを作成します
msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
ctproc = es1_CorrTerm(u'ユーザ使用率', "_USRUSE", record=ATCPU)
ctsysc = es1_CorrTerm(u'システム使用率', "_SYSUSE", record=ATCPU)
es1_add_corrmsg(ctproc, ctsysc, msgres)
```



(例 2) es1_CorrTermWithItem を用いた相関判定

```
msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
corrtproc = es1_CorrTerm(u'プロセッサ使用率',
                        value='_USRUSE + _SYSUSE', record=ATCPU)
corrtaaccu = es1_CorrTermWithItem(u'ユーザ$@のプロセッサ使用率', record=ATACCU,
                                itemkey=ATACCU.USRNAME, value=ATACCU.CPUUSE,
                                bases=ATCPU, item_name=u'ユーザ')
es1_add_corrmsg(corrtproc, corrtaaccu, msgres)
```



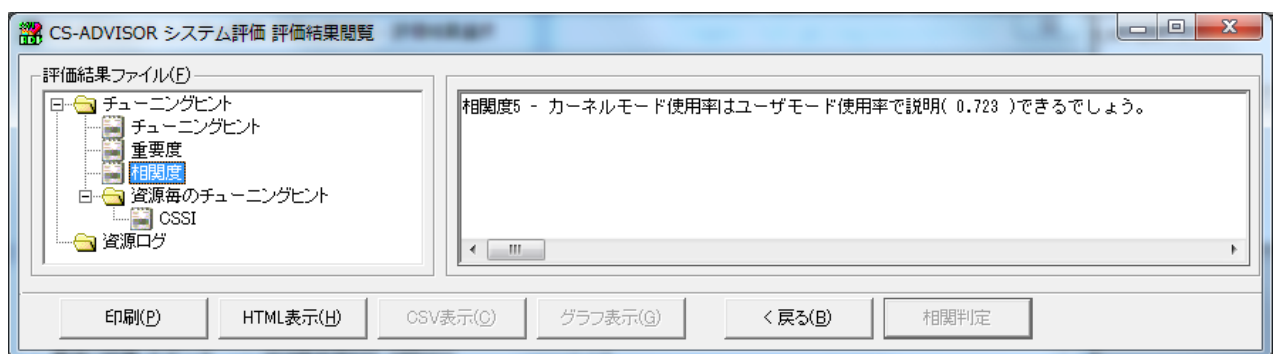
(例 3) es1_CorrTermByList を用いた相関判定

```
msgres = es1_get_msg_resource("CSSI", "x_list")

sql = u"select __TIME, _SYSUSE, _USRUSE from _ATCPU order by __TIME"
_sysuse = []
_usruse = []
for row in db.getcursor(sql):
    _sysuse.append( [ row[0], row[1] ] )
    _usruse.append( [ row[0], row[2] ] )

corr_sysuse = es1_CorrTermByList(u"カーネルモード使用率", _sysuse)
corr_usruse = es1_CorrTermByList(u"ユーザモード使用率", _usruse)

es1_add_corrmsg(corr_sysuse, corr_usruse, msgres)
```



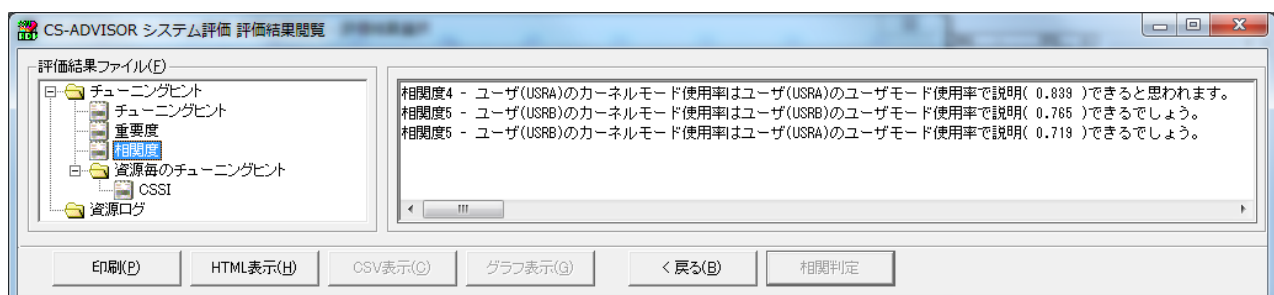
(例 4) es1_CorrTermWithItemByList を用いた相関判定

```
msgres = es1_get_msg_resource("CSSI", "x_list_withkey")

sql = u"select __TIME, _USERNAME, _SUM(_SYSUSE), _SUM(_USRUSE) from _ATACCD
group by _USERNAME, __TIME order by _USERNAME, __TIME"
_sysuse = []
_usruse = []
for row in db.getcursor(sql):
    _sysuse.append( [ row[0], row[1], row[2] ] )
    _usruse.append( [ row[0], row[1], row[3] ] )

corr_sysuse = es1_CorrTermWithItemByList(u"ユーザ$@のカーネルモード使用率", _sysuse)
corr_usruse = es1_CorrTermWithItemByList(u"ユーザ$@のユーザモード使用率", _usruse)

es1_add_corrmsg(corr_sysuse, corr_usruse, msgres)
```



2.5.5. 相関判定ナビゲーションとの連携

相関判定ナビゲーションでの相関判定対象項目を分類するグループを作成するために以下の手続きが使用可能です。これらの手続きは corr_main 関数からのみ呼び出せます。

es1_getNaviGroup(disptext, parent=None)

相関判定ナビゲーション画面で相関判定対象項目をグルーピングするためのオブジェクト (NaviGroup オブジェクト) を返します。

- disptext** : (省略不可／文字列型)
相関判定ナビゲーション画面で表示されるグループ名を示す文字列を指定してください。
- parent** : (デフォルト：None／NaviGroup オブジェクト)
相関判定ナビゲーション画面の左側のツリービューでグループを入れ子にしたい場合は parent に親のノードとなる NaviGroup オブジェクトを指定してください。

main 関数にて何らかの評価結果を出力している場合のみ、corr_main 関数は有効となります。

NaviGroup オブジェクトでは以下のメソッドが使用可能です。

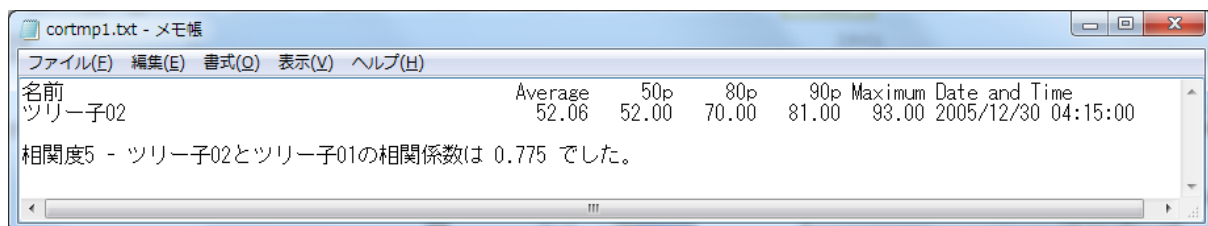
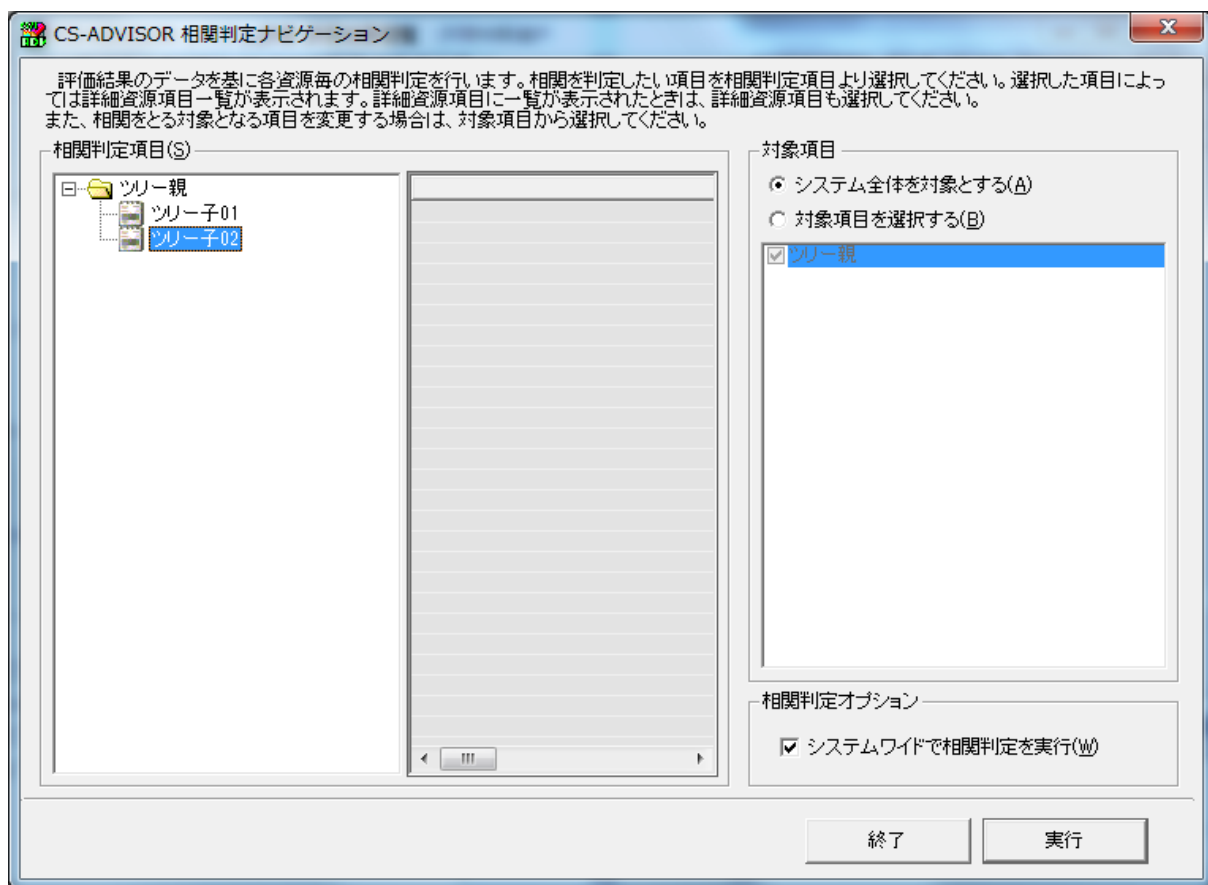
addNaviItem(disptext, value, record=None, where="", bases=(), msgtext="")

グループに属する相関判定対象項目を追加します。

- disptext** : (省略不可／文字列型)
相関判定ナビゲーション画面で表示されるグループ名を示す文字列を指定してください。
- value** : (省略不可／フィールドオブジェクト or 文字列型)
相関判定対象項目の値を示すフィールドオブジェクトか RDB の SQL 文の式に相当する文字列を指定してください。
- record** : (デフォルト：None／レコードオブジェクト or 文字列型)
相関判定対象項目を含むレコードオブジェクトか RDB のテーブルを示す文字列を指定してください。ただし、相関判定対象項目の値 (value 引数) をフィールドオブジェクトで指定した場合はこの引数を省略することが可能です (フィールドオブジェクトが属するレコードオブジェクトが使用されます)。
- where** : (デフォルト：""／文字列型)
相関判定対象項目のレコードの絞り込み条件となる SQL の条件式に相当する文字列を指定してください。
- bases** : (デフォルト：()／レコードオブジェクト or レコードオブジェクトのタプル (またはリスト))
レコードオブジェクトかレコードオブジェクトのタプル (またはリスト) を指定してください。bases を指定すると相関判定対象項目のレコードが存在せず bases に指定されたレコードが存在したインターバルの値を 0 として相関判定を行います。
- msgtext** : (デフォルト：""／文字列型)
相関判定ナビゲーションの実行結果で使用される相関判定対象項目を示す文字列を指定してください (msgtext が空文字列 ("") の場合は disptext が使用されます)。

```
def main (context, db, param):
    # ATCPU.USRUSE と ATCPU.SYSUSE で相関判定を行いメッセージを作成します
    msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
    ctproc = es1_CorrTerm(u'ユーザ使用率' , "_USRUSE" , record=ATCPU)
    ctsysc = es1_CorrTerm(u'システム使用率' , "_SYSUSE" , record=ATCPU)
    es1_add_corrmsg(ctproc, ctsysc, msgres)

def corr_main (context, db, param):
    # ATCPU.USRUSE と ATCPU.SYSUSE で相関判定ナビゲーションを行います
    ng = es1_getNaviGroup(u"ツリー親")
    ng.addNaviItem(u"ツリー子 01", "_USRUSE", ATCPU)
    ng.addNaviItem(u"ツリー子 02", "_SYSUSE", ATCPU)
```



es1_getNaviGroupWithKey(disptext, record, itemkey, keyfmt="", where="", bases=(), msgtext="", colname="", parent=None)

相関判定ナビゲーション画面で相関判定対象項目をグルーピングするためのオブジェクト（NaviGroupWithKey オブジェクト）を返します。NaviGroupWithKey オブジェクトに属する相関判定対象項目はキー項目を用いてグループ化されたそれぞれのインスタンス毎に相関判定が行われます。

- disptext** : (省略不可／文字列型)
相関判定ナビゲーション画面で表示されるグループ名を示す文字列を指定してください。
- record** : (省略不可／レコードオブジェクト or 文字列型)
相関判定対象項目を含むレコードオブジェクトか RDB のテーブルを示す文字列を指定してください。
- itemkey** : (省略不可／フィールドオブジェクト or 文字列型)
record で指定した表のレコード中でキーとなる項目を示すフィールドオブジェクトか RDB の SQL 文の式に相当する文字列を指定してください。キーとなる項目が複数ある場合はそれらをタプルかリストにまとめて指定してください。
- keyfmt** : (文字列型)
itemkey で指定した項目の値を書式化するための文字列を指定します。keyfmt 中に '\$@' を記述するとその部分が itemkey で指定した項目の文字列表現に置換されます。itemkey に複数の項目を指定した場合、置換は最も左側の '\$@' が itemkey[0] で置換され、次の '\$@' が itemkey[1] といった順番で行われます。keyfmt 引数に空文字列 ("") を指定した場合は各キー項目をカンマ (,) で区切った形で書式化されます。
- where** : (デフォルト: ""／文字列型)
相関判定対象項目のレコードの絞り込み条件となる SQL の条件式に相当する文字列を指定してください。
- bases** : (デフォルト: ()／レコードオブジェクト or レコードオブジェクトのタプル（またはリスト）)
レコードオブジェクトかレコードオブジェクトのタプル（またはリスト）を指定してください。bases を指定すると相関判定対象項目のレコードが存在せず bases に指定されたレコードが存在したインターバルの値を 0 として相関判定を行います。
- msgtext** : (デフォルト: ""／文字列型)
相関判定ナビゲーションの実行結果で使用するこのグループを示す文字列を指定してください（msgtext が空文字列 ("") の場合は disptext が使用されます）。
- colname** : (デフォルト: ""／文字列型)
相関判定ナビゲーション画面でインスタンスを選択するリストビューの列見出しとなる文字列を指定してください。
- parent** : (デフォルト: None／NaviGroup オブジェクト)
上記の es1_getNaviGroup と同様です。

main 関数にて何らかの評価結果を出力している場合のみ、corr_main 関数は有効となります。

NaviGroupWithKey オブジェクトでは以下のメソッドが使用可能です。

```
addNaviItem(disptext, value, msgtext="")
```

グループに属する相関判定対象項目を追加します。

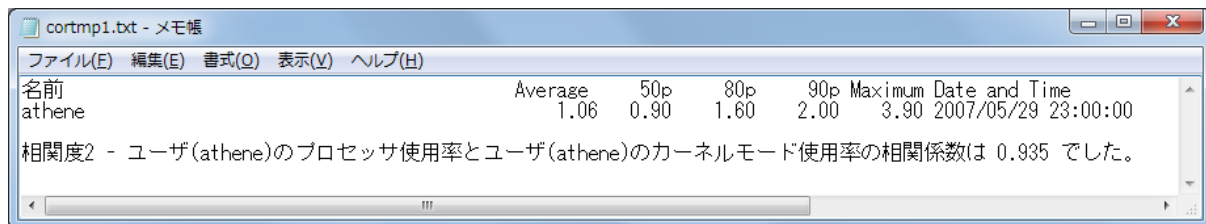
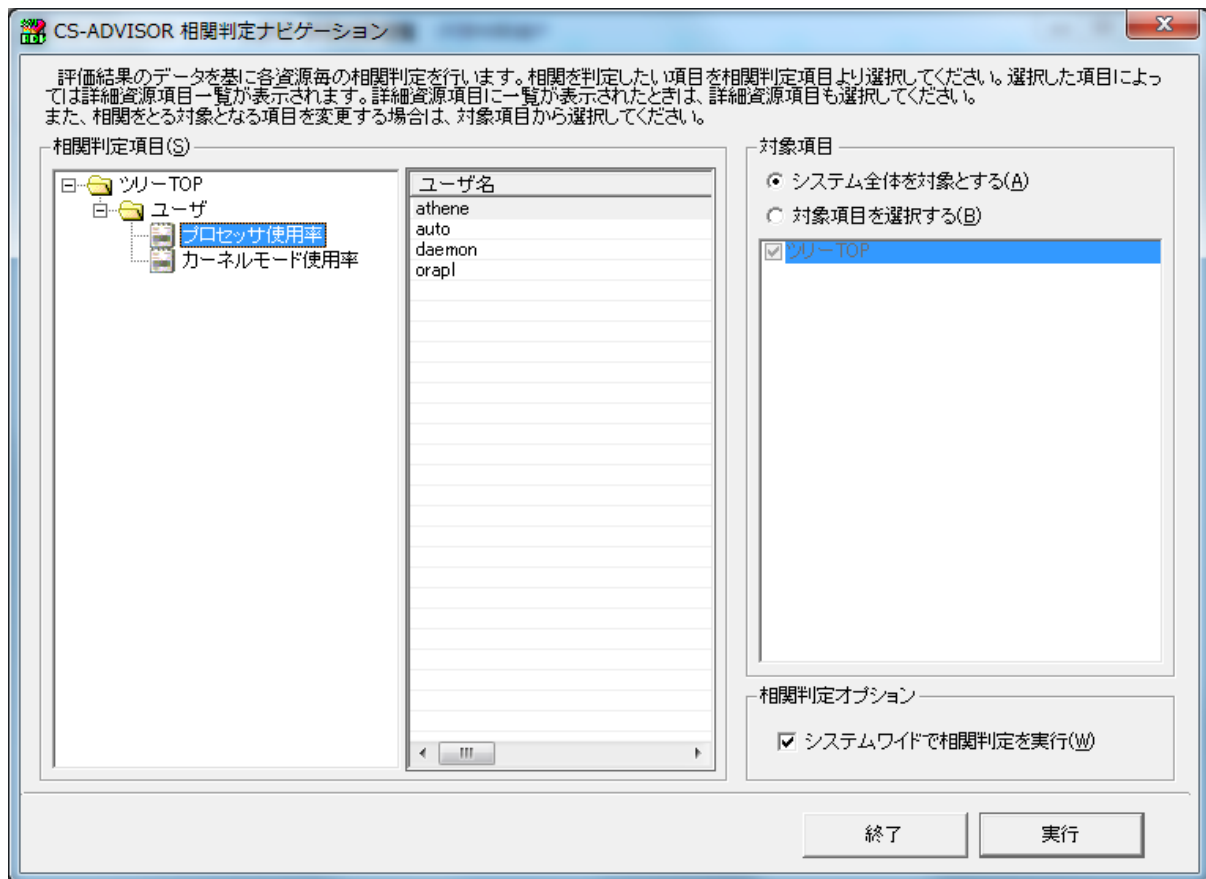
disptext : (省略不可／文字列型)
相関判定ナビゲーション画面で表示されるグループ名を示す文字列を指定してください。

value : (省略不可／フィールドオブジェクト or 文字列型)
相関判定対象項目の値を示すフィールドオブジェクトか RDB の SQL 文の式に相当する文字列を指定してください。value に指定する値は es1_getNaviGroupWithKey で record に指定した値と整合している必要があります。例えば es1_getNaviGroupWithKey で record に ATDEV (デバイス情報を示すレコードオブジェクト) を指定した場合は、value には ATDEV オブジェクトに属するフィールドオブジェクトか RDB のテーブル "_ATDEV" を使用して導き出される SQL の式を指定する必要があります。

msgtext : (デフォルト: ""／文字列型)
相関判定ナビゲーションの実行結果で使用される相関判定対象項目を示す文字列を指定してください (msgtext が空文字列 ("") の場合は disptext が使用されます)。

```
def main (context, db, param):
    msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
    ctproc = es1_CorrTerm(u'プロセスサ使用率' , "_CPUUSE" , record=ATACCU)
    ctsysc = es1_CorrTerm(u'カーネルモード使用率', "_SYSUSE" , record=ATACCU)
    es1_add_corrmsg(ctproc, ctsysc, msgres)

def corr_main (context, db, param):
    ngTop = es1_getNaviGroup(u'ツリ-TOP')
    ngLeaf = es1_getNaviGroupWithKey(u'ユーザ', record=ATACCU,
                                     itemkey=ATACCU.USERNAME,
                                     colname=u'ユーザ名', parent=ngTop)
    ngLeaf.addNaviItem(u'プロセスサ使用率', value=ATACCU.CPUUSE)
    ngLeaf.addNaviItem(u'カーネルモード使用率', value=ATACCU.SYSUSE)
```



es1_getNaviGroupByList(disptext, parent=None)

相関判定ナビゲーション画面で相関判定対象項目をグルーピングするためのオブジェクト（NaviGroupByList オブジェクト）を返します。db 中のレコードを対象とする NaviGroup オブジェクトと異なり、任意の値を指定することができます。

disptext : (省略不可／文字列型)
上記の es1_getNaviGroup と同様です。

parent : (デフォルト：None／NaviGroup オブジェクト)
上記の es1_getNaviGroup と同様です。

main 関数にて何らかの評価結果を出力している場合のみ、corr_main 関数は有効となります。

NaviGroupByList オブジェクトでは以下のメソッドが使用可能です。

`addNaviItem(disptext, values, bases=(), msgtext="")`

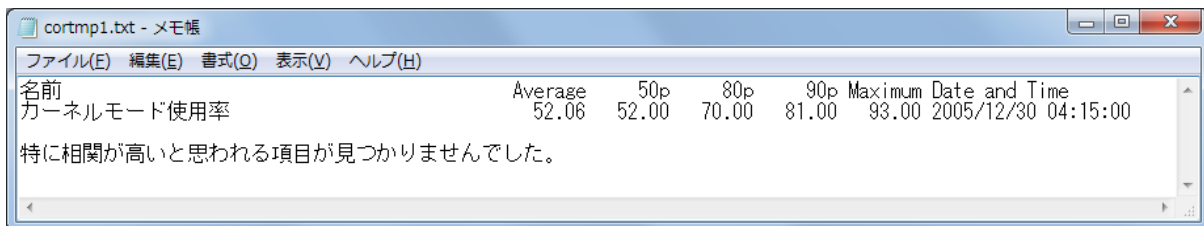
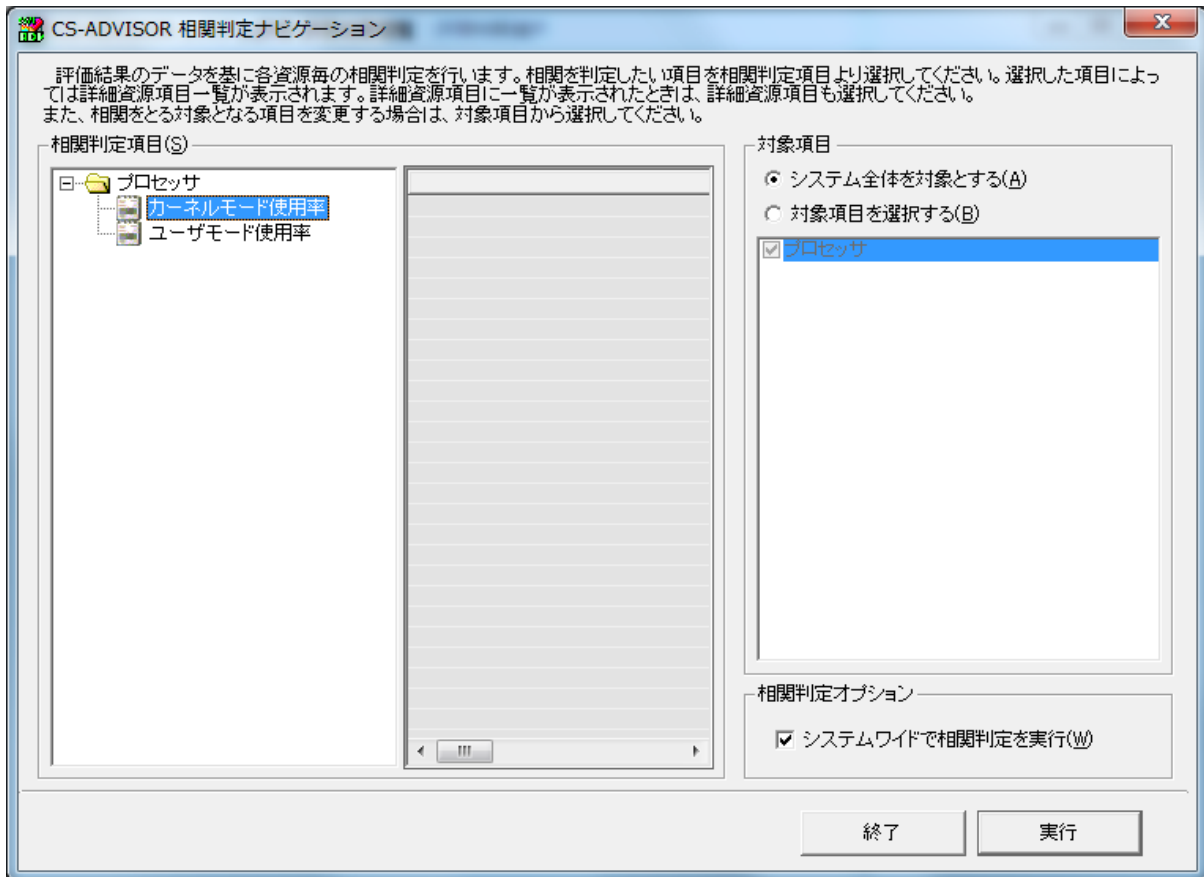
グループに属する相関判定対象項目を追加します。

- disptext** : (省略不可／文字列型)
相関判定ナビゲーション画面で表示されるグループ名を示す文字列を指定してください。
- values** : (省略不可／リスト)
相関判定対象項目の「時間」と「値」の 2 次元配列を指定します。[[time, value], ...]
time 1970/01/01 00:00 からの経過秒数
value 値
- bases** : (デフォルト：()／レコードオブジェクト or レコードオブジェクトのタプル（またはリスト）)
レコードオブジェクトかレコードオブジェクトのタプル(またはリスト)を指定してください。bases を指定すると相関判定対象項目のレコードが存在せず bases に指定されたレコードが存在したインターバルの値を 0 として相関判定を行います。
- msgtext** : (デフォルト：""／文字列型)
相関判定ナビゲーションの実行結果で使用される相関判定対象項目を示す文字列を指定してください（msgtext が空文字列（""）の場合は disptext が使用されます）。

```
def main (context, db, param):
    msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
    ctproc = es1_CorrTerm(u'プロセッサ使用率' , "_CPUUSE" , record=ATACCU)
    ctsysc = es1_CorrTerm(u'カーネルモード使用率', "_SYSUSE" , record=ATACCU)
    es1_add_corrmsg(ctproc, ctsysc, msgres)

def corr_main (context, db, param):
    ng = es1_getNaviGroupByList(u"プロセッサ")
    sql = "select __TIME, _SYSUSE, _USRUSE from _ATCPU order by __TIME"
    _sysuse = []
    _usruse = []
    for row in db.getcursor(sql):
        _sysuse.append( [row[0], row[1]] )
        _usruse.append( [row[0], row[2]] )

    ng.addNaviItem(u"カーネルモード使用率", _sysuse)
    ng.addNaviItem(u"ユーザモード使用率", _usruse)
```

es1_getNaviGroupWithKeyByList(disptext, bases=(), keyfmt="", msgtext="", colname="", parent=None)

相関判定ナビゲーション画面で相関判定対象項目をグルーピングするためのオブジェクト（NaviGroupWithKeyByList オブジェクト）を返します。db 中のレコードを対象とする NaviGroupWithKey オブジェクトと異なり、任意の値を指定することができます。

- disptext** : (省略不可／文字列型)
上記の es1_getNaviGroupWithKey と同様です。
- bases** : (デフォルト：())／レコードオブジェクト or レコードオブジェクトのタプル（またはリスト）
上記の es1_getNaviGroupWithKey と同様です。
- keyfmt** : (文字列型)
キー項目の値を書式化するための文字列を指定します。keyfmt 中に '\$@' を記述するとその部分がキー項目の文字列表現に置換されます。
- msgtext** : (デフォルト：""／文字列型)
上記の es1_getNaviGroupWithKey と同様です。

colname : (デフォルト: "" / 文字列型)
上記の es1_getNaviGroupWithKey と同様です。

parent : (デフォルト: None / NaviGroup オブジェクト)
上記の es1_getNaviGroup と同様です。

main 関数にて何らかの評価結果を出力している場合のみ、corr_main 関数は有効となります。

NaviGroupWithKeyByList オブジェクトでは以下のメソッドが使用可能です。

addNaviItem(disptext, values, msgtext="")
グループに属する相関判定対象項目を追加します。

disptext : (省略不可 / 文字列型)
相関判定ナビゲーション画面で表示されるグループ名を示す文字列を指定してください。

values : (省略不可 / リスト)
相関判定対象項目の「時間」と「キー」と「値」を 2 次元配列で指定します。 [[time, key, value], ...]
time 1970/01/01 00:00:00 からの経過秒数
key unicode 文字列
value 値

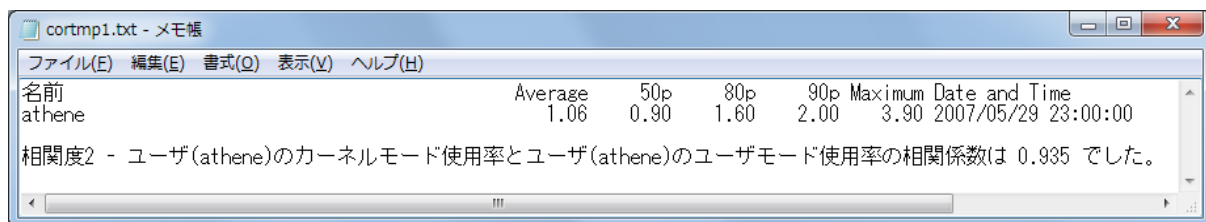
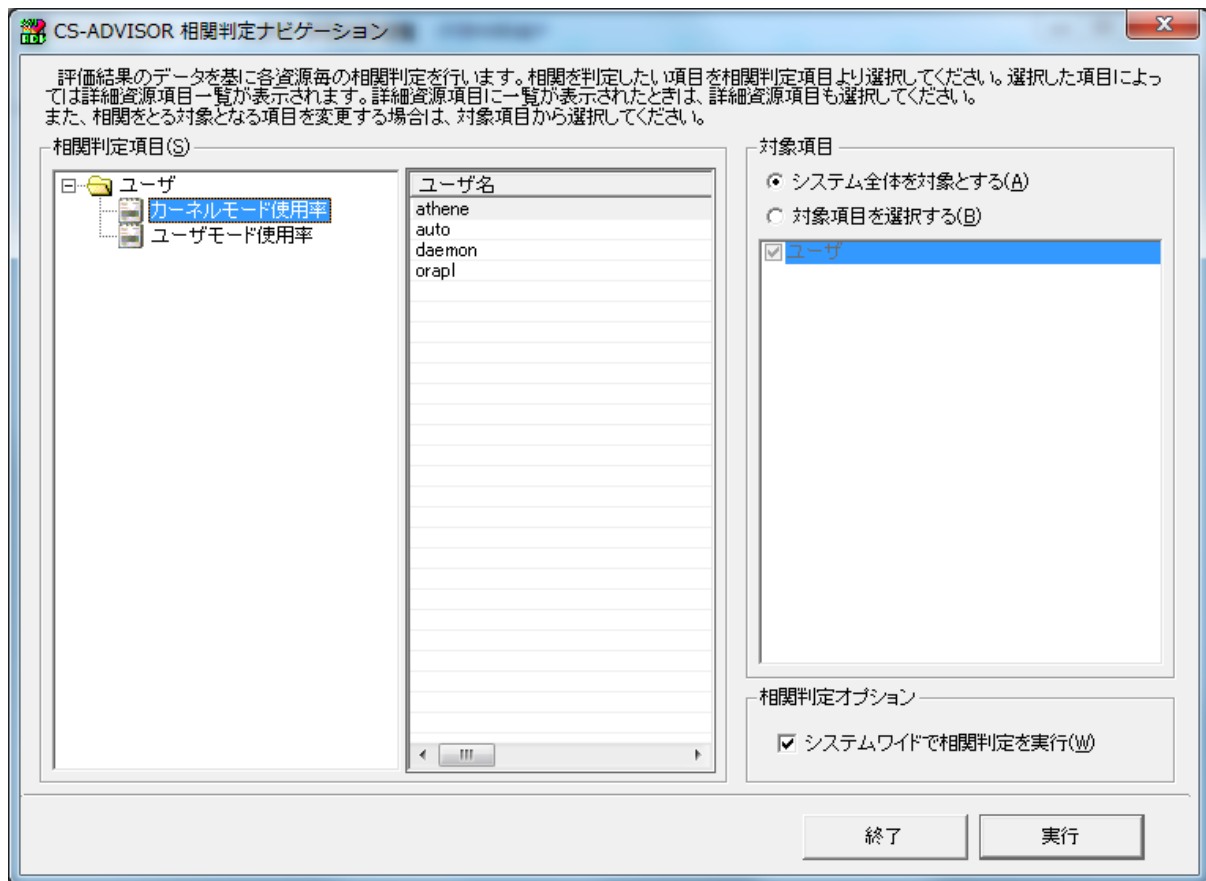
msgtext : (デフォルト: "" / 文字列型)
相関判定ナビゲーションの実行結果で使用される相関判定対象項目を示す文字列を指定してください (msgtext が空文字列 ("") の場合は disptext が使用されます)。

```
def main (context, db, param):
    msgres = es1_get_msg_resource('CSSI', filename='x_sample1')
    ctproc = es1_CorrTerm(u'プロセスサ使用率' , "_CPUUSE" , record=ATACCU)
    ctsysc = es1_CorrTerm(u'カーネルモード使用率', "_SYSUSE" , record=ATACCU)
    es1_add_corrmsg(ctproc, ctsysc, msgres)

def corr_main (context, db, param):
    ng = es1_getNaviGroupWithKeyByList(u"ユーザ", colname=u"ユーザ名")

    sql = "select __TIME, _USRNAME, _sum(_SYSUSE), _sum(_USRUSE) from _ATACCD
group by _USRNAME, __TIME order by _USRNAME, __TIME"
    _sysuse = []
    _usruse = []
    for row in db.getcursor(sql):
        _sysuse.append( [row[0], row[1], row[2]] )
        _usruse.append( [row[0], row[1], row[3]] )

    ng.addNaviItem(u"カーネルモード使用率", _sysuse)
    ng.addNaviItem(u"ユーザモード使用率", _usruse)
```



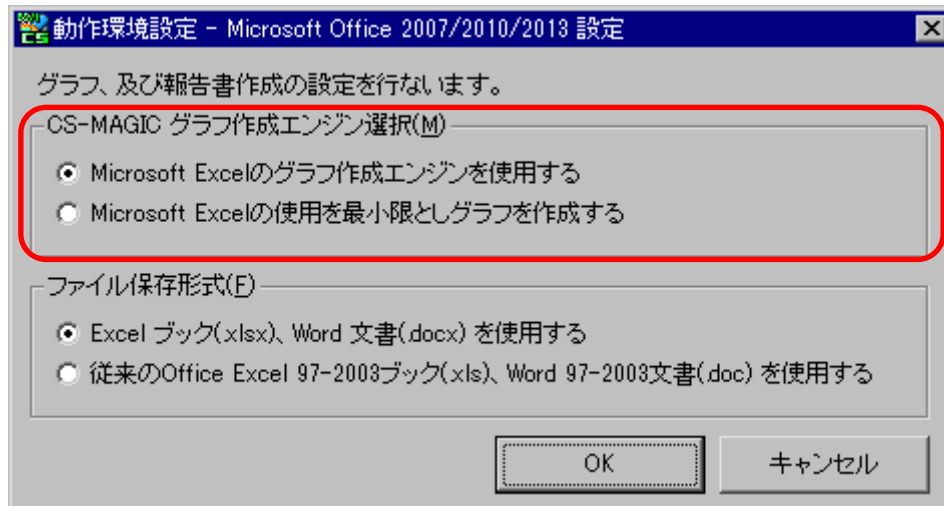
2.5.6. グラフ作成

拡張モジュールに以下の記述を行うことで excel ファイル形式、GIF/PNG ファイル形式のグラフを作成します。作成したグラフは CS-MAGIC のグラフと同様に Performance Web Service で閲覧することができます。

グラフの元データにフラットファイルや任意のデータを指定することで、ユーザ固有のグラフを生成することができます。

CSSI が使用するグラフ作成エンジンは 2 種類あり、

「環境(E)」メニューの「Microsoft Office 2007/2010/2013 設定(O)...」画面から指定します。



CS-MAGIC グラフ作成エンジン選択(M)

(1)Microsoft Excel のグラフ作成エンジンを使用する

Microsoft Excel を使用して Excel グラフファイルや GIF イメージ形式グラフを作成します。

(2)Microsoft Excel の使用を最小限としグラフを作成する

Microsoft Excel を極力使用せずに Excel グラフファイルや PNG イメージ形式グラフを作成します。

Microsoft Excel を起動しませんので、その分、グラフ作成時間が短くなります。

注意！

「Microsoft Excel の使用を最小限としグラフを作成する」を選択した場合、以下の制限事項があります。

- ・Microsoft .NET Framework4.5.2 の導入が必要です。
- ・Excel97-2003 ブック(.xls)は作成できません。
- ・モノクロハッチンググラフは作成できません。
- ・折れ線グラフの一部のマーカ背景色が透過ではなくプロットエリアの色になります(×、+、*)。
- ・積み上げ面グラフ(ラベル表示)のデータラベルが、系列名ではなく値になります。
- ・面、横棒、縦棒、折れ線、レーダー、散布図、複合グラフにデータテーブルの表示を指定した場合、Excel グラフには表示されますが、イメージ形式グラフには表示されません。

```
es1_get_plotdata(title="",pathname="",filename="",axisdata=[],type_y1=ES1_PLOT_TYPE_
LINE,type_y2=ES1_PLOT_TYPE_LINE,axis_x=AxisInf(),axis_y1=AxisInf(),axis_y2=AxisInf()
,width=554,height=310,legfontsize=9,titlefontsize=9,backcolor="white",option=ES1_PLOT_
OPT_Y1GRID,legend=ES1_PLOT_LEGEND_BOTTOM,htmlcsvpath="",pwsalign=None)
```

この手続きはグラフを作成するために必要な情報を保持するオブジェクト（PlotData オブジェクト）を返します。
PlotData オブジェクトの属性にファイル名、グラフの元データ、グラフ種類などの情報を設定します。

- title : (デフォルト: "" / 文字列型)
グラフ中に表示されるタイトル文字列を指定します。
- pathname : (デフォルト: "" / 文字列型)
グラフを作成するパスを指定します。
- filename : (デフォルト: "" / 文字列型)
グラフのファイル名(拡張子を除く)を示す文字列を指定してください。実際に作成されるファイル名はこ
で指定されたファイル名に拡張子 (".xls"/".xlsx"/".gif"/".png") を付けたものになります。
- axisdata : (デフォルト: [] / リスト型)
グラフの元データの系列をリストで指定します。リストの要素には es1_get_axisdata (後述) で生
成したオブジェクト (AxisData オブジェクト) のみ指定することができます。

(例) X 軸と Y1 軸を指定する場合

```
pd = es1_get_plotdata()
adx = es1_get_axisdata(axis=ES1_PLOT_AXIS_X)
ady = es1_get_axisdata(axis=ES1_PLOT_AXIS_Y1)
pd.axisdata = [adx, ady]
```

注意！

title、pathname、filename に「¥」で始まるエスケープシーケンスと「,」は使用できません。

type_y1 : (デフォルト: ES1_PLOT_TYPE_LINE / 以下より選択)
グラフの種類を指定します。

ES1_PLOT_TYPE_AREA : 面グラフ
ES1_PLOT_TYPE_STACKED_AREA : 面グラフ(積み上げを行う)
ES1_PLOT_TYPE_OVERLAP_AREA : 面グラフ(積み上げを行わない)
ES1_PLOT_TYPE_BAR : 集合横棒グラフ
ES1_PLOT_TYPE_STACKED_BAR : 横棒グラフ(積み上げを行う)
ES1_PLOT_TYPE_OVERLAP_BAR : 横棒グラフ(積み上げを行わない)
ES1_PLOT_TYPE_COLUMN : 集合縦棒グラフ
ES1_PLOT_TYPE_STACKED_COLUMN : 縦棒グラフ(積み上げを行う)
ES1_PLOT_TYPE_OVERLAP_COLUMN : 縦棒グラフ(積み上げを行わない)
ES1_PLOT_TYPE_LINE : データにマーカーが付けられた折れ線グラフ
ES1_PLOT_TYPE_PIE : 円グラフ
ES1_PLOT_TYPE_DOUGHNUT : ドーナツグラフ
ES1_PLOT_TYPE_RADAR : マーカー付きレーダーチャート
ES1_PLOT_TYPE_SCATTER : 散布図
ES1_PLOT_TYPE_SURFACE_TOP : 等高線グラフ(海水温グラフ)
ES1_PLOT_TYPE_SURFACE_REV : 色が逆順の等高線グラフ(海水温グラフ)

type_y2 : (デフォルト: ES1_PLOT_TYPE_LINE / 以下より選択)
複合グラフに使用する Y2 軸の種類を指定します。

ES1_PLOT_TYPE_AREA : 面グラフ
ES1_PLOT_TYPE_STACKED_AREA : 面グラフ(積み上げを行う)
ES1_PLOT_TYPE_OVERLAP_AREA : 面グラフ(積み上げを行わない)
ES1_PLOT_TYPE_COLUMN : 集合縦棒グラフ
ES1_PLOT_TYPE_STACKED_COLUMN : 縦棒グラフ(積み上げを行う)
ES1_PLOT_TYPE_OVERLAP_COLUMN : 縦棒グラフ(積み上げを行わない)
ES1_PLOT_TYPE_LINE : データにマーカーが付けられた折れ線グラフ

axis_x : (デフォルト: AxisInf() / AxisInf オブジェクト)
X 軸のラベルやピッチを指定します。指定可能な項目は「AxisInf オブジェクトの属性」を参照してください。

(例)X 軸ラベルに「時間」を指定する場合

```
pd = es1_get_plotdata()
pd.axis_x.label = u"時間"
```

(例)X 軸ラベルのフォントサイズに「12」を指定する場合

```
pd = es1_get_plotdata()
pd.axis_x.fontsize = "12"
```

axis_y1 : (デフォルト : AxisInf() / AxisInf オブジェクト)
Y1 軸のラベルやピッチを指定します。

axis_y2 : (デフォルト : AxisInf() / AxisInf オブジェクト)
Y2 軸のラベルやピッチを指定します。

AxisInf オブジェクトの属性

scalepitch

データ型 : 数値

値 : スケールのピッチを指定します。特に指定しない場合は、-1(自動)を指定します。

デフォルト : -1

scalemin

データ型 : 数値

値 : スケールの最小値を指定します。特に指定しない場合は、"E"(自動)を指定します。

デフォルト : E

scalemax

データ型 : 数値

値 : スケールの最大値を指定します。特に指定しない場合は、"E"(自動)を指定します。

デフォルト : E

label

データ型 : 文字列

値 : 軸のラベルを指定します。

デフォルト : ""

fontsize

データ型 : 数値

値 : 軸ラベルのフォントサイズをポイントで指定します。

デフォルト : 9

spacing

データ型 : 整数

値 : 項目軸目盛ラベルの間隔を指定します。

デフォルト : -1

rotation

データ型 : 整数

値 : 項目軸目盛ラベルの角度を指定します。

デフォルト : 90

format

データ型 : 文字列

値 : 軸目盛ラベルの書式文字列を指定します。

デフォルト : ""

option

データ型 : 選択

値 : 「ES1_PLOT_AXISMAX_BY_PITCH」を指定した場合、グラフ数値軸の最大値がピッチの倍数になります。
「ES1_PLOT_AXISLABEL_VERTICAL」を指定した場合、Y1/Y2 軸のラベルを縦書きに表示します。

デフォルト : 0

注意！

scalepitch,scalemin,scalemax,format は、axis_y1、axis_y2 に対して指定することが可能です。ただし、散布図を作成する場合は、axis_x に対しても指定することが可能です。

注意！

rotation,spacing は axis_x のみ指定可能です。

注意！

format は「Microsoft Excel のグラフ作成エンジンを使用する」を選択している場合のみ有効です。
format に「0.00」を指定すると Excel の制御で「0」と解釈されます。
セルの書式を変更する「'」を使用し、「'0.00」のように指定してください。

width : (デフォルト : 554 / 整数型)
グラフの幅をポイントで指定します。1 ポイントは 1/72 インチ。0.35mm。

height : (デフォルト : 310 / 整数型)
グラフの高さをポイントで指定します。1 ポイントは 1/72 インチ。0.35mm。

legfontsize : (デフォルト : 9 / 整数型)
凡例のフォントサイズをポイントで指定します。

titlefontsize : (デフォルト : 9 / 整数型)
グラフタイトルのフォントサイズをポイントで指定します。

backcolor : (デフォルト : white / 以下より選択)
プロットエリアの背景色を指定します。

white : 白色

gray : 灰色

option : (デフォルト : ES1_PLOT_OPT_Y1GRID / 以下より選択)
グラフの形状を指定します。複数の項目を指定することができます。

(例) データテーブルと Y1 軸のグリッド線を表示する場合

option = ES1_PLOT_OPT_DATATABLE | ES1_PLOT_OPT_Y1GRID

ES1_PLOT_OPT_DATATABLE	: データテーブルの表示(フォントサイズは後記 XFont キーで指定したサイズ)
ES1_PLOT_OPT_Y1GRID	: Y1 軸のグリッド線を表示する
ES1_PLOT_OPT_MONOCHROME	: モノクロにする ①
ES1_PLOT_OPT_USERCOLOR	: ユーザ指定の配色にする ②
ES1_PLOT_OPT_BOXPLOT	: 箱ひげ図を作成する ③

①と②は「ES/1 NEO CS シリーズ」メイン画面の「環境(E)」メニューにある「動作環境設定(共通)(E)...」において、「CSV/グラフオプション 1」タブにある次の設定に従います。詳細は「CS-MAGIC 使用者の手引き 7.3.2. 動作環境設定 (共通)」をご覧ください。

① モノクロハッチング

② 色の指定

③箱ひげ図は特殊なグラフのため以下のように指定してください。棒色は系列 1 と同色になります。凡例は表示できません。

option=ES1_PLOT_OPT_BOXPLOT

type_y1=ES1_PLOT_TYPE_STACKED_COLUMN

axisdata

- 0 番目(X 軸) : axis=ES1_PLOT_AXIS_X
- 1 番目(MIN-25PTL) : axis=ES1_PLOT_AXIS_NONE
- 2 番目(25PTL) : axis=ES1_PLOT_AXIS_Y1
- 3 番目(25PTL-MED) : axis=ES1_PLOT_AXIS_Y1
- 4 番目(MED-75PTL) : axis=ES1_PLOT_AXIS_Y1
- 5 番目(75PTL-MAX) : axis=ES1_PLOT_AXIS_NONE
- 6 番目～(省略可) : axis=ES1_PLOT_AXIS_NONE

legend : (デフォルト: ES1_PLOT_LEGEND_BOTTOM / 以下より選択)
凡例の表示位置を指定します。

ES1_PLOT_LEGEND_TOP : 上に凡例(凡例を表示する場合)
ES1_PLOT_LEGEND_BOTTOM : 下に凡例(凡例を表示する場合)
ES1_PLOT_LEGEND_LEFT : 左に凡例(凡例を表示する場合)
ES1_PLOT_LEGEND_RIGHT : 右に凡例(凡例を表示する場合)

htmlcsvpath : (省略可 / 文字列型)
グラフの HTML 数値ファイル、CSV 数値ファイルを出力するパスを指定します。
省略した場合は「CS-ADVISOR 評価条件ウィザード(4/6) CSV 形式ファイル」で指定したパスに出力します。

pwsalign : (省略可 / 以下より選択)
HTML 数値ファイルの各列の寄せ指定を行います。
実際の列数が PWSAlign の指定数よりも多い場合は、末尾の寄せ指定に従います。
省略した場合は寄せ指定を行いません。

L:左寄せ
C:中央揃え
R:右寄せ
N:指定無し

(例) PWSAlign=LLCR と指定した場合
1 列目と 2 列目を左寄せ、3 列目を中央揃え、4 列目を右寄せ

es1_get_axisdata(axis, label, type, declen=0, style=0, plottype=None, data=None)

この手続きはグラフに描画するための元データの系列や値を保持するオブジェクト（AxisData オブジェクト）を返します。系列毎にこの手続きを実行し、X 軸、Y1 軸等を設定します。

axis : (省略不可／以下より選択)
グラフのどの系列(X、Y1、Y2 軸)に使用されるかを指定します。

ES1_PLOT_AXIS_NONE : データシートには出力するがグラフに使用しない
ES1_PLOT_AXIS_X : X 軸
ES1_PLOT_AXIS_Y1 : Y1 軸
ES1_PLOT_AXIS_Y2 : Y2 軸

label : (省略不可／文字列型)
系列の名前を指定します。ここで指定した内容はデータシートに出力されます。

type : (デフォルト：str／以下より選択)
元データの型を指定します。後述する data 属性に設定した値をフォーマットします。

int : 整数型
long : 長整数型
float : 浮動小数点型
unicode : unicode 文字列型
str : 文字列型

declen : (デフォルト：0／整数型)
元データの小数点以下の桁数を指定します。

style : (デフォルト：ES1_PLOT_STYLE_LINE_THIN／以下より選択)
折れ線のスタイルを指定します。

ES1_PLOT_STYLE_LINE_HAIR : 折れ線の太さ - 極細
ES1_PLOT_STYLE_LINE_THIN : 折れ線の太さ- 細
ES1_PLOT_STYLE_LINE_MEDIUM : 折れ線の太さ- 中
ES1_PLOT_STYLE_LINE_THICK : 折れ線の太さ- 太
ES1_PLOT_STYLE_LINE_NOMARKER : 折れ線上のマーカ-を消す
ES1_PLOT_STYLE_LINE_HIDE : 折れ線を消す

(例) 折れ線の太さを中、マーカ-を消す場合

style = ES1_PLOT_STYLE_LINE_MEDIUM | ES1_PLOT_STYLE_LINE_NOMARKER

メモ！

ES1_PLOT_STYLE_LINE_HIDE は、「Microsoft Excel の使用を最小限としグラフを作成する」を選択している場合のみ有効です。

plottype : (None／以下より選択)
同軸に複数種類のグラフを描画する場合、系列毎にグラフの種類を指定します。

ES1_PLOT_TYPE_AREA : 面グラフ
 ES1_PLOT_TYPE_STACKED_AREA : 面グラフ(積み上げを行う)
 ES1_PLOT_TYPE_OVERLAP_AREA : 面グラフ(積み上げを行わない)
 ES1_PLOT_TYPE_COLUMN : 集合縦棒グラフ
 ES1_PLOT_TYPE_STACKED_COLUMN : 縦棒グラフ(積み上げを行う)
 ES1_PLOT_TYPE_OVERLAP_COLUMN : 縦棒グラフ(積み上げを行わない)
 ES1_PLOT_TYPE_LINE : データにマーカーが付けられた折れ線グラフ

(例) Y1 軸の系列 1 に折れ線、系列 2 に縦棒のグラフを作成する場合

```
pd = es1_get_plotdata()
ady1 = es1_get_axisdata(axis=ES1_PLOT_AXIS_Y1, label=u"系列 1", plottype=ES1_PLOT_TYPE_LINE)
ady2 = es1_get_axisdata(axis=ES1_PLOT_AXIS_Y1, label=u"系列 2", plottype=ES1_PLOT_TYPE_COLUMN)
pd.axisdata = [ady1, ady2]
```

data : (デフォルト : None／リスト型)
グラフに描画する元データをリストで指定します。ここで指定した内容はデータシートに出力されます。

(例) Y1 軸に 1～10 の値を使用する場合

```
ady = es1_get_axisdata(axis=ES1_PLOT_AXIS_Y1,
label="sample",
type=int)

ady.data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

(例) DB オブジェクトを利用した時間 (HH:MM) の抽出

```
adx = es1_get_axisdata(axis=ES1_PLOT_AXIS_X,
label="time",
type=str)

for r in db.getRecordIter(ATCPU):
ymdhm = time.strftime("%Y/%m/%d %H:%M", time.localtime(r.time))
ymd,hm = ymdhm.split(' ')
adx.data.append(hm)
```

(例) DB オブジェクトを利用したプロセッサのカーネルモード使用率の抽出

```
ady = es1_get_axisdata(axis=ES1_PLOT_AXIS_Y1,
label="SYSUSE",
type=float,
declen=2)

for r in db.getRecordIter(ATCPU):
ady.data.append(r[ATCPU.SYSUSE])
```

es1_output_file(pd, ordernum=None, pwsinfo=None, excel=ES1_OUTPUT_OPTION_NONE, gif=ES1_OUTPUT_OPTION_NONE, html=ES1_OUTPUT_OPTION_NONE, csv=ES1_OUTPUT_OPTION_NONE, pws=ES1_OUTPUT_OPTION_NONE, engine=ES1_OUTPUT_ENGINE_NONE, timeout=300):

この手続きは es1_get_plotdata で作成したグラフの情報を Excel 形式、GIF/PNG 形式、HTML 形式、CSV 形式で出力します。EXCEL/GIF ファイルを生成する es1_make_excelplot、Performance Web Service 連携用ファイルを生成する es1_link_pws の機能を 1 つの手続きで行なうことができます。ファイル出力にはこの手続きを使用してください。

- pd** : (省略不可／PlotData オブジェクト)
es1_get_plotdata で作成した PlotData オブジェクトを指定します。
- ordernum** : (省略不可／整数型)
Performance Web Service 閲覧機能で表示されるグラフを識別する一意の番号を指定します。指定可能な数字は $1 < \text{ordernum} < 9999999$ です。グラフ毎に異なる番号を指定してください。Performance Web Service 閲覧画面では ordernum に指定された番号に 20000000 を加算した番号が表示されます。
- pwsinfo** : (省略可／es1_pwsinfo オブジェクト)
Performance Web Service 連携用の xml ファイルの情報を保持する es1_pwsinfo オブジェクト（後述）を指定します。
- excel** : (デフォルト：ES1_OUTPUT_OPTION_NONE／以下より選択)
es1_get_plotdata で作成した PlotData オブジェクトを使用して Excel 形式のグラフを生成します。指定可能な値は以下です。

ES1_OUTPUT_OPTION_FALSE : 評価条件ファイルのオプションに関係なく、ファイルを出力しません。
ES1_OUTPUT_OPTION_TRUE : 評価条件ファイルのオプションに関係なく、ファイルを出力します。
ES1_OUTPUT_OPTION_NONE : 評価条件ファイルのオプションに従い、ファイルを出力します。

- gif** : (デフォルト：ES1_OUTPUT_OPTION_NONE／excel 参照)
es1_get_plotdata で作成した PlotData オブジェクトを使用して GIF/PNG 形式のグラフを生成します。
「Microsoft Excel のグラフ作成エンジンを使用する」を選択した場合は GIF 形式グラフ、「Microsoft Excel の使用を最小限としグラフを作成する」を選択した場合は PNG 形式グラフを作成します。指定可能な値は excel と同様です。
- html** : (デフォルト：ES1_OUTPUT_OPTION_NONE／excel 参照)
es1_get_plotdata で作成した PlotData オブジェクトを使用して HTML 形式の数値情報を生成します。指定可能な値は excel と同様です。
- csv** : (デフォルト：ES1_OUTPUT_OPTION_NONE／excel 参照)
es1_get_plotdata で作成した PlotData オブジェクトを使用して CSV 形式の数値情報を生成します。指定可能な値は excel と同様です。

pws : (デフォルト : ES1_OUTPUT_OPTION_NONE / excel 参照)
GIF 形式のグラフと HTML 形式の数値情報を Performance Web Service にアップロードするための xml ファイルを生成します。
指定可能な値は excel と同様です。

注意 !

評価条件ファイル作成時に出力しない設定としているオプションに対して、ES1_OUTPUT_OPTION_TRUE を選択した場合、ファイルの出力先フォルダを指定してください。
excel, gif ファイルを出力する場合、es1_get_plotdata の pathname 引数を指定してください。
html, csv ファイルを出力する場合、es1_get_plotdata の htmlcsvpath 引数を指定してください。

engine : (デフォルト : ES1_OUTPUT_ENGINE_NONE / 以下より選択)
グラフ作成エンジンを指定します。指定可能な値は以下です。

ES1_OUTPUT_ENGINE_NONE : 「グラフ作成エンジン選択」の選択に従います。
ES1_OUTPUT_ENGINE_EXCEL : 「Microsoft Excel のグラフ作成エンジンを使用する」
ES1_OUTPUT_ENGINE_OPENXML : 「Microsoft Excel の使用を最小限としグラフを作成する」

timeout : (デフォルト : 300 / 整数型)
グラフ作成のタイムアウト時間を秒数で指定します。

メモ !

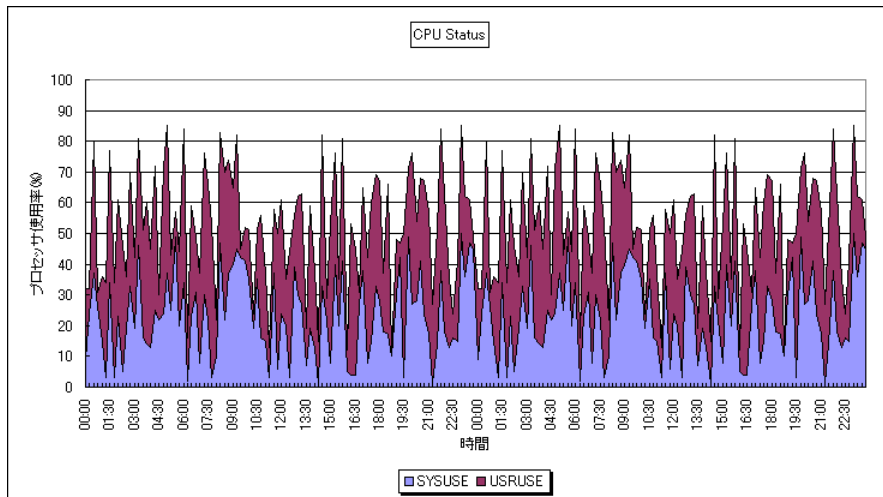
timeout は、「Microsoft Excel の使用を最小限としグラフを作成する」を選択している場合のみ有効です。

戻り値 : (成功時 : ES1_OUTPUT_RESULT_OK)
グラフ作成中にエラーした場合、WARN レベルのログを出力し、処理を継続します。
戻り値を確認することで、明示的に処理を停止することができます。

(例) PlotData オブジェクトの情報を GIF、HTML 形式で出力する場合

```
pd = es1_get_plotdata()
pwsinfo = es1_pwsinfo()
rt = es1_output_file(pd, ordernum=1, pwsinfo=pwsinfo, gif=ES1_OUTPUT_OPTION_TRUE,
html=ES1_OUTPUT_OPTION_TRUE)
if rt != ES1_OUTPUT_RESULT_OK:
    es1_exec_error(u'グラフの作成に失敗しました。')
```

(例) es1_output_file(pd)



es1_make_excelplot (PlotData オブジェクト)

この手続きは es1_get_plotdata で作成した PlotData オブジェクトを使用してグラフを生成します。
PlotData オブジェクトに必要な情報をセットした後、この手続きを行ってください。

注意！

この手続きは非推奨です。es1_output_file 手続きを使用してください。

**es1_link_pws(pd, resource_title=u"その他", stime=None, etime=None,
target=None, plot_title=plotdata.title, xaxis_type=0, plot_id=None)**

この手続きは es1_make_excelplot で作成したグラフを Performance Web Service にアップロードするために必要な xml ファイルを生成します。引数の内容は es1_output_file 手続きと同じです。

注意！

この手続きは非推奨です。es1_output_file 手続きを使用してください。

es1_pwsinfo(resource_title=es1_const.RESOURCE_OTHER, stime=None, etime=None, target=None, plot_title=None, xaxis_type=ES1_XAXISTYPE_NONE, plot_id=None)

この手続きは Performance Web Service 連携用の xml ファイルの情報を保持するオブジェクト(es1_pwsinfo オブジェクト)を返します。

resource_title : (デフォルト: u"その他"/以下から選択)
Performance Web Service 閲覧機能の「Performance Information」画面で「資源カテゴリ・タブ」に表示される資源カテゴリ名を指定します。「ES/1 NEO CS シリーズ」メイン画面、「環境(E)」メニュー、「ES/1 NEO Performance Web Service 連携(W)...」、「ES/1 NEO Performance Web Service クエリーカテゴリ」、「カテゴリ設定」に表示されるカテゴリ名のみ指定することができます。

(例) 資源カテゴリの「プロセッサ」タブに表示する場合
es1_link_pws(pd, 1, resource_title = u"プロセッサ")

stime : (省略可/整数型)
開始日時を示す 1970/01/01 00:00 からの経過秒数を指定します。
指定を行わない場合は CS-ADVISOR の評価条件作成時に指定した開始日時が指定されます。
(例) 2006/01/01 12:00 を指定する場合
stime = time.mktime((2006, 1, 1, 12, 0, 0, 0, 0, -1))

etime : (省略可/整数型)
終了日時を示す 1970/01/01 00:00 からの経過秒数を指定します。
指定を行わない場合は CS-ADVISOR の評価条件作成時に指定した終了日時が指定されます。

target : (省略可/タプル型 or 文字列型)
Performance Web Service データベースに登録するサイト/システム名を指定します。
指定した内容によって単一システムグラフ、複数システムグラフ(同一サイト内)、複数システムグラフ(複数サイト)のカテゴリに分類されます。

target の指定	登録されるカテゴリ	PWS プレビュー
タプル型 ("siteA", "systemA")	単一システムグラフ	siteA/systemA
文字列型 "siteA"	複数システムグラフ (同一サイト内)	siteA/siteA_複数システム*1
空文字列 ""	複数システムグラフ (複数サイト)	複数サイト/複数サイト_複数システム*2

*1
「複数システム」、「複数サイト」は「Performance Web Service Uploader」の設定によって自動的に付加されます。詳細は「Performance Web Service 使用者の手引き」「2.3.2.3. オプション」の「サイト/システムの自動登録」をご覧ください。

省略した場合は es1_link_pws の呼び出された場所によって以下のように設定されます。

main_init/main_term 関数内 : 評価条件中に指定されたすべてのシステムが同一サイトの場合はサイト名の文字列、異なるサイトを含む場合は空文字列
main 関数内 : main 関数実行中のシステムのタプル*2

*2

複数のシステムを対象に CS-ADVISOR を実行した場合、システム毎に main 関数が実行されます。

plot_title : (デフォルト : plotdata.title / 文字列型)
Performance Web Service 管理者設定機能の「グラフ表示順設定」画面に表示されるグラフタイトルを指定します。
省略した場合はグラフ中に表示されるタイトル文字列が表示されます (PlotData オブジェクトの title 属性で指定した値)。

注意！

グラフファイルの生成は es1_make_excelplot にて行います。Performance Web Service にて閲覧するためには es1_make_excelplot と es1_link_pws の手続きが必要です。

xaxis_type : (デフォルト : ES1_XAXISTYPE_NONE / 以下より選択)
Performance Web Service 閲覧機能の「Performance Information」画面に表示される際の時系列を指定します。

ES1_XAXISTYPE_NONE : 作成したグラフの期間により自動設定
ES1_XAXISTYPE_DET : 詳細タブ
ES1_XAXISTYPE_WEEK : 週次タブ
ES1_XAXISTYPE_MONTH : 月次タブ
ES1_XAXISTYPE_YEAR : 年次タブ

plot_id : (デフォルト : plot_title / 文字列型)
Performance Web Service 閲覧機能の「Performance Information」画面に表示されるグラフタイトルを指定します。
省略した場合は plot_title 属性で指定した値になります。

2.5.7. メール送信

拡張モジュールを利用してメール送信を行うことができます。宛先、件名、本文を任意に指定することができます。メール送信は、以下の手順で行います。

- (1)メール設定ファイルの作成
- (2)Es1Mail オブジェクトの作成
- (3)Es1Mail オブジェクトの sendmail メソッドの呼び出し

(1)メール設定ファイル

メールサーバや宛先など、メール送信に必要な情報を記述するテキストファイルです。
ファイル名は任意に指定することができます。文字コードは SHIFT-JIS、改行は CR-LF にします。

メモ！

行頭が # で始まる行はコメントとして認識されます。

・メールサーバ設定

メール送信に使用するメールサーバを設定します。
[:MAILSERVER:]の行を記述し、各項目を追加します。

```
[ :MAILSERVER: ]
server=mail.co.jp
port=25
timeout=30
from_address=tokyo@co.jp
```

指定可能な項目は以下になります。

キー	省略可/不可	型	デフォルト値	説明
server=	不可	文字列	""	メールサーバを指定
port=	可	数値	25	ポート番号を指定
timeout=	可	数値	30	タイムアウト値（秒）を指定
from_address=	不可	文字列	""	メール送信元を指定
user=	可	文字列	""	認証時のユーザ名を指定
password=	可	文字列	""	認証時のパスワードを指定
starttls=	可	0 or 1	0	暗号化通信（STARTTLS）を使用する場合は 1 を指定

・宛先設定

メール送信用の宛先を指定します。グループとそのグループに含まれる宛先を指定します。
グループの指定は[グループ名]で行い、グループに含まれる宛先の情報を記述します。

```
[東京]
  東京 01,tokyo01@co.jp,メモ 1
  東京 02,tokyo02@co.jp,メモ 2
[大阪]
  大阪 01,osaka01@co.jp,メモ 1
  大阪 02,osaka02@co.jp,メモ 2
```

#この例では東京と大阪の 2 つのグループが登録されます。

1 つの宛先は"名前,メールアドレス,メモ"の形式で行います。"メモ"は省略可能です。

メモ！
グループ名はすべて大文字で登録されます。

(2)Es1Mail オブジェクト

メール設定ファイルの読み込み、メール送信、宛先情報の参照を行います。

`es1_get_es1mail(filename=u"")`

この手続きはメール送信用のオブジェクト(Es1Mail オブジェクト)を返します。

filename : (デフォルト : CS インストールフォルダ¥es1mail.ini／文字列)
メール設定ファイルをフルパスで指定します。

Es1Mail オブジェクトのメソッド

`get_address(group=u"")`

メール設定ファイルの宛先設定に登録した情報をディクショナリで返します。

group : (デフォルト : u""／unicode 文字列)
メール設定ファイルに指定した宛先設定のグループ名を指定します。指定した場合は該当グループの情報を、省略した場合はすべてのグループの情報をディクショナリで返します。グループが存在しない場合は空のディクショナリを返します。

ディクショナリは次の形式です。

※グループを省略した場合

```
{ グループ名:
  {名前:
    {"address":メールアドレス, "memo":メモ},
  },
}
```

※グループを指定した場合

```
{名前:
  {"address":メールアドレス, "memo":メモ},
}
```

`get_address_list(group=u"")`

メール設定ファイルの宛先設定に登録したメールアドレスの情報をリストで返します。

group : (デフォルト: u"" / unicode 文字列)
メール設定ファイルに指定した宛先設定のグループ名を指定します。指定した場合は該当グループのメールアドレスを、省略した場合はすべてのグループのメールアドレスをリストで返します。グループが存在しない場合は空のリストを返します。

`sendmail(to, subject, body)`

メールを送信します。メール送信時にエラーをハンドリングできるように、専用の Exception クラス (ES1_Mail_Exception クラス) を用意しています。

to : (省略不可 / リストかタプル)
メールの送信先を指定します。

subject : (省略不可 / unicode 文字列)
メールの件名を指定します。

body : (省略不可 / unicode 文字列)
メールの本文を指定します。

ES1_Mail_Exception クラス

メール送信時のエラーメッセージ、発信元となるオリジナルのクラスを特定できるように以下の属性を用意しています。

errbase : 発信元のクラスオブジェクト。

errmsg : エラーメッセージ。

```
def main(context, db, param):
    # メール設定ファイルの読み込み、Es1Mail オブジェクトの作成
    email = es1_get_es1mail()
    # 宛先の取得
    to = email.get_address_list()
    # メール送信
    try:
        email.sendmail(to, u"件名", u"本文")
    except ES1_Mail_Exception, e:
        # エラーメッセージをログに出力
        es1_logwarn(u"%s %s", e.errbase.__class__.__name__, e.errmsg)
```

2.6. CS 標準提供処理が使用するグループ名

2.6.1. 重要度メッセージ、相関度メッセージを分類するグループ名一覧

カーネル
プロセッサ
警告
メモリー
I/O サブシステム
コマンド
ユーザ
ネットワーク
Oracle
SQL Server
Symfoware
DB2
SAP ERP
ネットワーク回線
MIB ノード
HTTP ログ
TCP セッション
TCP PtoP
IBM i
プロセッサ(ESX ホスト)
プロセッサ(仮想マシン)
メモリー(ESX ホスト)
メモリー(仮想マシン)
I/O サブシステム(ESX ホスト)
I/O サブシステム(仮想マシン)
ネットワーク(ESX ホスト)
ネットワーク(仮想マシン)
データストア
WebSphere (サーバ名)
WebLogic(サーバ名)
z/VM
MySQL
Interstage(ワークユニット名)
JBoss
プロセッサ(Hyper-V)
メモリー(Hyper-V)
ストレージ(Hyper-V)
ネットワーク(Hyper-V)
プロセッサ(パーティション)
Tomcat
NetApp プロセッサ
NetApp WAFL
NetApp ボリューム

2.6.2. 数値データを分類するグループ名一覧

カーネル
プロセッサ
警告
メモリー
バッファキャッシュ
ファイルアクセス
I/O サブシステム
ファイルシステム
コマンド
ユーザ
ユーザコマンド
ネットワーク
Oracle サマリー
Oracle セッション毎のプロセッサ
Oracle セッション毎のメモリー
Oracle セッション毎のデータベースアクセス
Oracle セッション毎のバッファ
Oracle セッション毎のスキャン数
Oracle セッション毎のロングスキャン数
Oracle セッション毎のスキャンブロック数
Oracle セッション毎のスキャン行数
Oracle セッション毎の Redo サイズ
Oracle セッション毎の Redo 待ち時間
Oracle データファイル毎のアクセス
Oracle ドライブ毎のアクセス
Oracle テーブル
SQL Server サマリー
SQL Server のログファイル
SQL Server のユーザ
SQL Server のキャッシュ
SQL Server のデータベース
Symfoware サマリー
Symfoware ユーザ毎のプロセッサ
Symfoware ユーザ毎の SQL 実行
Symfoware ユーザ毎のメモリー
Symfoware ユーザ毎のロック
Symfoware ユーザ毎のレコードアクセス
Symfoware のバッファ
Symfoware のデータベース
DB2 サマリー
DB2 バッファ
DB2 アプリケーション
DB2 ユーザ
DB2 テーブルスペース
DB2 テーブル
SAP ERP サマリー

SAP ERP トランザクション
SAP ERP ワークプロセス
SAP ERP タスクタイプ
SAP ERP ユーザ
SAP ERP 端末
SAP ERP トランザクションコード
SAP ERP プログラム
ネットワーク回線サマリ
ネットワーク回線ポート
ネットワーク回線サーバ
ネットワーク回線クライアント
ネットワーク回線相手先
ネットワーク回線サーバ・ポート
ネットワーク回線クライアント・ポート
ネットワーク回線相手先ポート
MIB サイト
MIB ノード
MIB インタフェース
HTTP サービス全体
URL
HTTP サーバ
HTTP クライアント
TCP セッション・サマリー
TCP セッション・送受信量
TCP セッション・送受信データ量
TCP セッション・送受信データセグメント数
TCP セッション・再送受信データセグメント数
TCP セッション・再送受信データセグメント比率
TCP セッション・再送受信データ量
TCP セッション・再送受信データ比率
TCP セッション・接続回数
TCP セッション・ウィンドウサイズゼロ
TCP セッション・レスポンス時間
TCP セッション・回線遅延時間
TCP セッション・要求送信時間
TCP セッション・処理時間
TCP セッション・応答送信時間
TCP セッション・処理回数
TCP セッション・セッション確立に要した時間
TCP セッション・チューニングヒントデータ
TCP PtoP・送受信量
TCP PtoP・送信量
TCP PtoP・再送信データ量
TCP PtoP・再送信データ比率
TCP PtoP・送信セグメント数
TCP PtoP・再送信データセグメント数
TCP PtoP・再送信データセグメント比率
TCP PtoP・ウィンドウサイズゼロ送信回数

TCP PtoP・回線遅延時間
TCP PtoP・チューニングヒントデータ
環境変更履歴
IBM i の構成情報
IBM i の概要
IBM i のインターバルサマリー
IBM i のプロセッサ
IBM i のディスク
IBM i のディスク(ASP 別)
IBM i のプール
IBM i のプール(プール別)
IBM i のジョブ
IBM i のジョブ(サブシステム別)
IBM i のジョブ(プール別)
プロセッサ(データセンタ)
プロセッサ(ESX ホスト)
プロセッサ・物理(仮想マシン)
プロセッサ・仮想(仮想マシン)
メモリー(データセンタ)
メモリー・使用量(ESX ホスト)
メモリー・Swap(ESX ホスト)
メモリー・Balloon(ESX ホスト)
メモリー・使用量(仮想マシン)
メモリー・Swap(仮想マシン)
メモリー・Balloon(仮想マシン)
I/O サブシステム(ESX ホスト)
I/O サブシステム(仮想マシン)
ネットワーク(ESX ホスト)
ネットワーク(仮想マシン)
データストア
マイグレーション
WebSphere (サーバ名)
WebLogic(サーバ名)
z/VM 構成情報
z/VM インターバル・サマリー情報
z/VM 実行効率情報
z/VM 論理分割プロセッサ使用状況
z/VM ユーザ別プロセッサ使用状況
z/VM ユーザ別主記憶使用状況
z/VM ユーザ・ページング状況
z/VM スケジューラ情報
z/VM ゲスト・サマリー情報
z/VM ゲスト稼働状況
MySQL 設定
MySQL メモリ
MySQL InnoDB
MySQL MyISAM
MySQL アクティビティ

MySQL 接続

Interstage(ワークユニット名)

JBoss

構成表・基本(Hyper-V)

構成表・基本(パーティション)

プロセッサ(全体)

プロセッサ(論理)

プロセッサ(パーティション)

メモリー(全体)

メモリー(ダイナミックメモリー)

ストレージ(物理ディスク)

ストレージ(論理ディスク)

ストレージ(仮想 IDE コントローラ)

ストレージ(仮想ストレージデバイス)

ネットワーク(ルートパーティション)

ネットワーク(仮想スイッチ)

ネットワーク(仮想ネットワークアダプタ)

ネットワーク(レガシーネットワークアダプタ)

ネットワーク(仮想スイッチポート)

Tomcat

NetApp 構成表

NetApp システム

NetApp プロセッサ

NetApp WAFL

NetApp アグリゲート

NetApp ボリューム

2.7. その他の es1lib モジュールの属性

2.7.1. es1_setenv オブジェクト

このオブジェクトは以下の属性を持ちます。

is_x64

実行環境により、True(64bit)か、False(32bit)のいずれかの値を持ちます。

cs_path

CS インストールフォルダのフルパスです。デフォルトインストールの場合、C:¥IIM¥CS になります。

cssi_path

cssi フォルダのフルパスです。デフォルトインストールの場合、C:¥IIM¥CS¥cssi になります。

iimwork_path

IIM_WORK フォルダのフルパスです。デフォルトインストールの場合、C:¥IIM_WORK になります。

csout_path

CSOUT フォルダのフルパスです。デフォルトインストールの場合、C:¥IIM_DATA¥CS¥CSOUT になります。

graphout_path

GRAPHOUT フォルダのフルパスです。デフォルトインストールの場合、C:¥IIM_DATA¥CS¥GRAPHOUT になります。

xaxislf

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブで「軸ラベル(D)-項目軸ラベルの改行をする」の指定により、True(する)か False(しない)のいずれかの値を持ちます。

xaxisweek

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブで「軸ラベル(D)-月次グラフの日付に曜日を付加」の指定により、ES1_AXISWEEK_NONE(しない)、ES1_AXISWEEK_JP(日本語表記)、ES1_AXISWEEK_EN(英語表記)のいずれかの値を持ちます。

cpunumset

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブの「CPU 搭載数(P)」の指定により、True(自動設定)か False(常に 1 を設定)のいずれかの値を持ちます。

windisk

CS 動作環境設定(共通)の「CSV/グラフオプション 2」タブで「Windows デバイス情報(W)」の指定により、ES1_WINDISK_ALL(すべて読み込む)、ES1_WINDISK_LOGICAL(論理デバイスのみ読み込む)、ES1_WINDISK_PHYSICAL(物理デバイスのみ読み込む)のいずれかの値を持ちます。

skip_unix_dev

CS 動作環境設定(共通)の「CSV/グラフオプション 2」タブで「Unix アンダーバーデバイス(U)」の指定により、True(読み込まない)か False(読み込む)のいずれかの値を持ちます。

nulltozero

CS 動作環境設定(共通)の「CSV/グラフオプション 2」タブで「欠損値出力方法(Z)」の指定により、True(ゼロを出力する)か False(欠損値を出力する)のいずれかの値を持ちます。

topnum

CS 動作環境設定(共通)の「CSV/グラフオプション 2」タブで「デバイス、コマンド情報等の表示制限(N)」の指定により、0(すべて出力する)か指定された集約数のいずれかの値を持ちます。

sumunit

CS 動作環境設定(共通)の「CSV/グラフオプション 2」タブで「時間による集約(I)」の指定により、0(集約しない)か指定された時間のいずれかの値を持ちます。

orsessid_proc

CS 動作環境設定(共通)の「Oracle 設定」タブで「Oracle セッション ID(H)-Oracle セッション ID 毎に処理する」の指定により、True(する)か False(しない)のいずれかの値を持ちます。

sql65pid_proc

CS 動作環境設定(共通)の「SQL Server 設定」タブで「SQL Server 6.5 PID(L)-ユーザを PID 毎に処理する」の指定により、True(する)か False(しない)のいずれかの値を持ちます。

udblogin_proc

CS 動作環境設定(共通)の「DB2 設定」タブで「DB2 アプリケーション名(4)-アプリケーション名の処理にユーザのログイン ID を含める」の指定により、True(する)か False(しない)のいずれかの値を持ちます。

lowactproc_read

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「評価オプション」タブで「稼働率の低いプロセス情報の読み込み(P)」の指定により、True(読み込む)か False(読み込まない)のいずれかの値を持ちます。

padinterval

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「評価オプション」タブで「欠損インターバルの埋め込み(I)」の指定により、True(埋め込む)か False(埋め込まない)のいずれかの値を持ちます。

sort

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「評価オプション」タブで「ソートの方法(S)」の指定により、ES1_SORT_BY_NUM(数値でソート)か ES1_SORT_BY_STR(名前でソート)のいずれかの値を持ちます。

sortcsv

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「評価オプション」タブで「ソートの方法(S)-CSV、グラフもここでの指定でソートする」の指定により、ES1_SORT_DEFAULT(ソートしない)、ES1_SORT_BY_NUM(数値でソート)、ES1_SORT_BY_STR(名前でソート)のいずれかの値を持ちます。

このオブジェクトは以下のメソッドを持ちます。

get_cpumnt()

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブで「CPU 搭載数(P)」の指定により、現在処理中のシステムの CPU 搭載数設定値を返します。

get_memsize()

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブで「物理メモリ量/スワップメモリ量」の指定により、現在処理中のシステムの物理メモリ量設定値(設定値が無い場合は-1)を返します。

get_swpsize()

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブで「物理メモリ量/スワップメモリ量」の指定により、現在処理中のシステムのスワップメモリ量設定値(設定値が無い場合は-1)を返します。

get_physvol()

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「対象デバイス指定」タブの指定により、現在処理中のシステムの評価対象とする物理デバイス名からなるタプル(存在しない場合は None)を返します。

get_phyevol()

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「対象デバイス指定」タブの指定により、現在処理中のシステムの評価除外とする物理デバイス名からなるタプル(存在しない場合は None)を返します。

get_logsvol()

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「対象デバイス指定」タブの指定により、現在処理中のシステムの評価対象とする論理デバイス名からなるタプル(存在しない場合は None)を返します。

get_logevol()

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「対象デバイス指定」タブの指定により、現在処理中のシステムの評価除外とする論理デバイス名からなるタプル(存在しない場合は None)を返します。

get_sorsess()

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「Oracle セッション指定」タブの指定により、現在処理中のシステムの評価対象とする Oracle セッションからなるタプル(存在しない場合は None)を返します。タプルの各要素は Oracle ユーザ名と OS ユーザ名の 2 要素からなるタプルです。

get_eorsess()

CS 動作環境設定(CS-ADVISOR / CS-Network ADVISOR)の「Oracle セッション指定」タブの指定により、現在処理中のシステムの評価除外とする Oracle セッションからなるタプル(存在しない場合は None)を返します。タプルの各要素は Oracle ユーザ名と OS ユーザ名の 2 要素からなるタプルです。

fmt_filename(filename)

CS 動作環境設定(共通)の「CSV/グラフオプション 1」タブで「出力ファイル名の形式(F)」に指定された形式でフォーマットした文字列を返します。評価対象システムの種類によって「サイト/システム名」項目の置換文字列が異なります。

単一システム	: 評価対象システムのサイト名/システム名
複数システム(同一サイト内)	: 評価対象システム群のサイト名
複数システム(複数サイト)	: 空文字列

filename : (省略不可/文字列型)

グラフファイル名。グラフ作成以外にも使用することができます。

(例) pd = es1_get_plotdata()

pd.filename = es1_setenv.fmt_filename("面グラフ")

注意！

`fmt_filename` メソッドが `main` 関数から実行された場合、評価対象システムが複数システム(同一サイト内)、複数システム(複数サイト)であっても `main` 関数実行中のシステムのサイト/システム名に置換されます。

2.7.2. `es1_timebase`

評価条件ファイル作成時に指定した区切り時刻の値を持ちます。

2.7.3. `es1_strftime(ts, fmt=u"%Y/%m/%d-%H:%M", timebase=None)`

時間を区切り時刻に対応した文字列で返します。

- `ts` : (省略不可/基準時 (1970/01/01 00:00:00) からの経過秒数を示す整数値)
文字列への変換対象とする時間。None が指定された場合は `u"....../..-...."` を返します。
- `fmt` : (デフォルト: `u"%Y/%m/%d-%H:%M"` / unicode 文字列)
書式指定文字列。この指定に従い `ts` を文字列に変換します。`fmt` に指定可能な文字列は以下です。

文字列	意味
<code>%d</code>	10 進数で表す月の日付 (01~31)
<code>%H</code>	24 時間表記の時間 (00~23)
<code>%I</code>	12 時間表記の時間 (01~12)
<code>%j</code>	10 進数で表す年初からの日数 (001~366)
<code>%m</code>	10 進数で表す月 (01~12)
<code>%M</code>	10 進数で表す分 (00~59)
<code>%p</code>	実行環境における午前/午後を表す文字列
<code>%U</code>	10 進数で表す週の通し番号。日曜日を週の最初の日とする (00~53)
<code>%w</code>	10 進数で表す曜日 (0~6、日曜日が 0)
<code>%W</code>	10 進数で表す週の通し番号。月曜日を週の最初の日とする (00~53)
<code>%y</code>	10 進数で表す西暦の下 2 桁 (00~99)
<code>%Y</code>	10 進数で表す 4 桁の西暦
<code>%%</code>	パーセント記号

注意！

CSSI では `fmt` の文字列に制限を設けていません。python の `strftime` の仕様に従います。

- `timebase` : (デフォルト: None/int)
区切り時刻を指定します。指定可能な値は 0-23 の整数です。省略した場合は評価条件ファイル作成時に指定した区切り時刻を用いて変換します。

2.7.4. es1_ptl(L, pos)

単一のパーセンタイル値を求めます。

L : (省略不可／数値型のリスト)
パーセンタイル値を求めるリスト

pos : (省略不可／浮動小数点型)
0 以上 1 以下の浮動小数点を指定してください。pos で指定されたパーセンタイル値を返します。

```
# 80 パーセンタイルを取得
intArray = []
intArray.append(10)
intArray.append(30)
intArray.append(50)
intArray.append(70)
p80 = es1_ptl(intArray, 0.8) # 80 パーセンタイル値である「58.0」が取得されます
```

2.7.5. es1_ptls(L, *posl)

複数のパーセンタイル値を求めます。

L : (省略不可／数値型のリスト)
パーセンタイル値を求めるリスト

pos l : (省略不可／浮動小数点型)
0 以上 1 以下の浮動小数点を指定してください。posl で指定されたパーセンタイル値のリストを返します。

```
# 50、80、90 パーセンタイルを取得
intArray = []
intArray.append(10)
intArray.append(30)
intArray.append(50)
intArray.append(70)
ptls = es1_ptls(intArray, 0.5, 0.8, 0.9)
p50 = ptls[0] # 50 パーセンタイル値である「40.0」が取得されます
p80 = ptls[1] # 80 パーセンタイル値である「58.0」が取得されます
p90 = ptls[2] # 90 パーセンタイル値である「64.0」が取得されます
```


2.7.6. es1_plus(values)

欠損値を除いた計算結果を返します。values に指定したすべての要素の合計値を返します。ただし、values の中で 1 つでも 0 未満の要素が存在する場合は-1 を返します。

values : (省略不可／リスト or タプル)
計算対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_plus(values) # result は-1 となります

# 欠損値を含まない場合
values = [1, 2, 3, 4]
result = es1_plus(values) # result は 10 となります
```

2.7.7. es1_sum(values)

欠損値を除いた計算結果を返します。values に指定したすべての要素の合計値を返します。ただし、0 未満の要素は対象外となります。また、values に指定したすべての要素が 0 未満であった場合は-1 を返します。

values : (省略不可／リスト or タプル)
計算対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_sum(values) # result は 6 となります

# 欠損値を含まない場合
values = [1, 2, 3, 4]
result = es1_sum(values) # result は 10 となります
```

2.7.8. es1_minus(values)

欠損値を除いた計算結果を返します。values の 1 個目の要素が 0 未満の場合は -1 を、そうでなければ 1 個目の要素から 2 個目以降の要素を引いた値を返します。2 個目以降の要素のうち、0 未満の要素は対象外となります。

values : (省略不可/リスト or タプル)
計算対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
#欠損値を含む場合
values = [10, 5, 3, -1]
result = es1_minus(values) # result は 2 となります

#欠損値を含まない場合
values = [10, 5, 3, 1]
result = es1_minus(values) # result は 1 となります
```

2.7.9. es1_multi(values)

欠損値を除いた計算結果を返します。values に指定した要素を乗算した結果を返します。values の中で 0 未満の要素が 1 つでも存在する場合は -1 を返します。

values : (省略不可/リスト or タプル)
計算対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
#欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_multi(values) # result は -1 となります

#欠損値を含まない場合
values = [1, 2, 3, 4]
result = es1_multi(values) # result は 24 となります
```

2.7.10. es1_div(numer, denom)

欠損値を除いた計算結果を返します。numer を denom で割った値を返します。ただし numer が 0 未満か denom が 0 未満の場合は -1 を、denom が 0 の場合は 0 を返します。

numer : (省略不可／数値)
分子。int/float/numpy.number 型以外の要素は欠損値とします。

denom : (省略不可／数値)
分母。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
numer = 50
denom = -1
result = es1_div(numer, denom) # result は -1 となります

# 欠損値を含まない場合
numer = 50
denom = 100
result = es1_div(numer, denom) # result は 0.5 となります
```

2.7.11. es1_cnt(values)

欠損値を除いた計算結果を返します。values に指定した要素の中で 0 以上の要素の個数を返します。

values : (省略不可／リスト or タプル)
対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_cnt(values) # result は 3 となります

# 欠損値を含まない場合
values = [1, 2, 3, 4]
result = es1_cnt(values) # result は 4 となります
```

2.7.12. es1_avg(values)

欠損値を除いた計算結果を返します。values に指定したすべての要素の平均値を返します。ただし、0 未満の要素は対象外となります。また、values に指定したすべての要素が 0 未満であった場合は-1 を返します。

values : (省略不可/リスト or タプル)
計算対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_avg(values) # result は 2 となります

# 欠損値を含まない場合
```

2.7.13. es1_min(values)

欠損値を除いた計算結果を返します。values に指定したすべての要素の最小値を返します。ただし、0 未満の要素は対象外となります。また、values に指定したすべての要素が 0 未満であった場合は-1 を返します。

values : (省略不可/リスト or タプル)
対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_min(values) # result は 1 となります

# 欠損値を含まない場合
values = [1, 2, 3, 4]
result = es1_min(values) # result は 1 となります
```

2.7.14. es1_max(values)

欠損値を除いた計算結果を返します。values に指定したすべての要素の最大値を返します。ただし、0 未満の要素は対象外となります。また、values に指定したすべての要素が 0 未満であった場合は-1 を返します。

values : (省略不可/リスト or タプル)
対象とする数値のリスト(タプル)。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
values = [1, 2, 3, -1]
result = es1_max(values) # result は 3 となります

# 欠損値を含まない場合
values = [1, 2, 3, 4]
result = es1_max(values) # result は 4 となります
```

2.7.15. es1_ratio(number, denom, over=100)

欠損値を除いた計算結果を返します。denom に対する、numer の割合を百分率で返します。ただし numer が 0 未満か denom が 0 以下の場合は-1 を返します。

- numer : (省略不可／数値)
割合を求める値。int/float/numpy.number 型以外の要素は欠損値とします。
- denom : (省略不可／数値)
基準となる値。int/float/numpy.number 型以外の要素は欠損値とします。
- over : (デフォルト：100／数値)
割合の閾値。割合が over に指定された値を超えた場合は-1 を返します。over が負の値の場合は、閾値の確認を行いません。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
numer = 50
denom = -1
result = es1_ratio(numer, denom) # result は-1 となります

# 欠損値を含まない場合
numer = 25
denom = 100
result = es1_ratio(numer, denom) # result は 25 となります
```

2.7.16. es1_ratio_rev(number, denom)

欠損値を除いた計算結果を返します。denom に対する、denom と numer の差分の割合を百分率で返します。ただし numer が 0 未満か denom が 0 以下の場合は-1 を返します。また、numer が denom より大きい場合も-1 を返します。

- numer : (省略不可／数値)
基準となる値と割合を求める値の差分。int/float/numpy.number 型以外の要素は欠損値とします。
- denom : (省略不可／数値)
基準となる値。int/float/numpy.number 型以外の要素は欠損値とします。

```
# 欠損値を含む場合
numer = 50
denom = -1
result = es1_ratio_rev(numer, denom) # result は-1 となります

# 欠損値を含まない場合
numer = 25
denom = 100
result = es1_ratio_rev(numer, denom) # result は 75 となります
```

2.7.17. es1_is_pvm_os(db)

処理対象システムが AIX の Power VM を使用しているか Bool 値で返します。

db : (省略不可)
db オブジェクトを指定してください。

2.7.18. es1_is_instance_profile()

評価条件ファイルにインスタンスプロファイルが設定されているか Bool 値で返します。

基本的に、インスタンスプロファイルに含まれるインスタンスのみを対象として評価を行います。

インスタンスプロファイルの詳細については、「プロファイル機能 使用者の手引き」を参照してください。

注意！

インスタンスプロファイルを参照する関数は、main のみ実行可能です。

main_init/main_term/corr_main 呼び出し時は実行できません。

2.7.19. es1_get_oracle_profile(disptext="")

Oracle のインスタンスプロファイルを展開した結果を namedtuple のリストで返します。インスタンスプロファイルを利用していない場合、もしくは展開結果が 0 件の場合、空のリストを返します。namedtuple は、以下の形式です。

type_name : InstanceProfile
field_name : DOMAIN (ドメイン名)
 : DBNAME (データベース名)

disptext : (省略可)
DOMAIN、DBNAME を使った表示用文字列を指定します。書式は format 関数に準じます。
disptext を指定した場合、namedtuple の最終フィールドに「DISPTTEXT」を追加します。

```
#disptext を省略した場合
for instance in es1_get_oracle_profile():
    print instance
    #InstanceProfile(DOMAIN=u'DOMAIN1', DBNAME=u'DBNAME1')

#disptext を指定した場合
for instance in es1_get_oracle_profile(disptext=u"{0}/{1}"):
    print instance
    #InstanceProfile(DOMAIN=u'DOMAIN1' DBNAME=u'DBNAME1',
                     DISPTTEXT=u'DOMAIN1/DBNAME1')
```

2.7.20. es1_get_symfoware_profile(disptext="")

Symfowareのインスタンスプロファイルを展開した結果をnamedtupleのリストで返します。インスタンスプロファイルを利用していない場合、もしくは展開結果が0件の場合、空のリストを返します。namedtupleは、以下の形式です。

type_name : InstanceProfile
field_name : DBNAME (RDB システム名)

disptext : (省略可)
上記、es1_get_oracle_profileと同様です。

2.7.21. es1_get_sqlserver_profile(disptext="")

SQL Serverのインスタンスプロファイルを展開した結果をnamedtupleのリストで返します。インスタンスプロファイルを利用していない場合、もしくは展開結果が0件の場合、空のリストを返します。namedtupleは、以下の形式です。

type_name : InstanceProfile
field_name : INST (インスタンス名)

disptext : (省略可)
上記、es1_get_oracle_profileと同様です。

2.7.22. es1_get_db2_profile(disptext="")

DB2 のインスタンスプロファイルを展開した結果を namedtuple のリストで返します。インスタンスプロファイルを利用していない場合、もしくは展開結果が0件の場合、空のリストを返します。namedtupleは、以下の形式です。

type_name : InstanceProfile
field_name : INST (ノード名)
: DBNAME (データベース別名)

disptext : (省略可)
上記、es1_get_oracle_profileと同様です。

2.7.23. es1_get_saperp_profile(disptext="")

SAP ERPのインスタンスプロファイルを展開した結果をnamedtupleのリストで返します。インスタンスプロファイルを利用していない場合、もしくは展開結果が0件の場合、空のリストを返します。namedtupleは、以下の形式です。

type_name : InstanceProfile
field_name : INST (インスタンス名)

disptext : (省略可)
上記、es1_get_oracle_profileと同様です。

2.7.24. es1_exec_error(msg)、es1_exec_cancel(msg)

GUI での評価実行時にはメッセージボックスに msg を表示し、バッチでの評価実行時にはログに msg を出力し、評価処理を中止します。es1_exec_cancel では警告レベルのログが、es1_exec_error では停止レベルのログが出力されます。

msg : (省略不可／文字列型)
メッセージに出力する文字列を指定してください。

```
es1_exec_cancel(u'キャンセルします')
```



```
es1_exec_error (u'中断します')
```



2.7.25. es1_loginfo(msg)、es1_logwarn(msg)

msg : (省略不可／文字列型)
ログに出力する文字列を指定してください。

バッチでの評価実行時にログに msg を出力します。es1_loginfo では情報レベルのログが、es1_logwarn では警告レベルのログが出力されます。

2.8. レコード/フィールドオブジェクトの例外規定

(1)SAP ERP トランザクション情報 (R3TRN) については以下のフィールドを使用することができません。

R3TRN.STDATE(開始日付)
R3TRN.STTIME(開始時刻-HHMM)
R3TRN.STTIMES(開始時刻-SS)
R3TRN.ENDDATE(終了日付)
R3TRN.ENDTIME(終了時刻-HHMM)
R3TRN.ENDTIMES(終了時刻-SS)

SAP ERP トランザクション情報の開始日時と終了日時を参照するには上記フィールドのかわりに以下のフィールドを使用してください。

R3TRN.STARTTM(開始日時)
R3TRN.ENDTM(終了日時)

どちらのフィールドも基準時 (1970/01/01 00 : 00 : 00) からの経過秒数を示す整数値で表現されています。

(2)名称に英小文字が使用されている表 (例 : WASEntityEJB) についてはすべて英大文字 (例 : WASENTITYEJB) で記述してください。

(3)Flatfile Maintenance データ集約関連の各レコードオブジェクトについては TDACTIVESEC (一月当たりの総秒数) を使用することができません。TDACTIVESEC に相当する値を取得するにはインターバル長を示す "___INTVL"フィールドを使用してください。

以下の表についてはインターバル長 ("___INTVL"フィールド、SimpleRecord オブジェクトの intvl 属性) は「-1」となります。

ATFSS
ORSESS
R3DB02
R3ST22

2.9. esutil モジュール

es1util モジュールは CS-ADVISOR の評価結果に関連しない、雑多な機能を提供します。このモジュールを使用するには拡張モジュール中に

```
from es1util import <手続き名>
```

というインポート宣言を記述します。es1util が公開している属性の名はすべて"es1_"または"ES1_"というプレフィックスで始まっています。名前の衝突を避けるために拡張モジュールではすべて英大文字のオブジェクト名（レコードオブジェクトと一致の可能性）や"es1_"または"ES1_"で始まるオブジェクト名は使用しないようにしてください。

以下に es1util が提供する機能を機能別に示します。

2.9.1. イベントログの作成

es1_eventlog (message, eventId, eventType=ES1_EVENTLOG_ERROR_TYPE)

この手続きは Windows のイベントログを作成します。

message : (省略不可 / unicode 文字列)
イベントログに出力するメッセージを指定します。

eventId : (省略不可 / 1～9 の整数)
イベント ID を指定します。

eventType : (デフォルト : ES1_EVENTLOG_ERROR_TYPE / 以下から選択)
イベントログの種類を指定します。

ES1_EVENTLOG_ERROR_TYPE	: エラーイベント
ES1_EVENTLOG_WARNING_TYPE	: 警告イベント
ES1_EVENTLOG_INFORMATION_TYPE	: 情報イベント
ES1_EVENTLOG_AUDIT_SUCCESS	: 成功の監査イベント
ES1_EVENTLOG_AUDIT_FAILURE	: 失敗の監査イベント