VCL のコンストラクタ・デストラクタ

これは VCL の初期化時・終了時に呼び出されるもので vcl_init/vcl_fini アクションとは関係有りません。
正規表現のコンパイルや VMOD のロードなどを行なっています。

ここまでの変換された VCL の元

ユーザが記述した VCL とデフォルトの VCL が記述されています。

VCL の設定

Varnish が処理の際に使う VCL の設定が入っています。

Varnish のソースを読む

一つ一つの記述を VCL で書いてみて C に変換して確認するのは非常に手間がかかります。
かと言って Varnish のソースをすべて見て理解するのは非常に難しいです。
また、あまり望ましくはないですが内部の関数を巧みに使いトリッキーなことをするにも、高度な処理を行う上で Varnish の動きも最低限把握する必要があります。
そこでポイントとなるソースと読み方について解説します。

/lib/libvcl/generate.py

VCL で利用する各種アクションの戻り値や変数の一覧や型など非常に重要な内容が記述されています。このファイルは以下の内容を含みます。

VCL のトークン一覧

tokens に使用可能な演算子などが定義されています。

アクション(vcl_recv など)で利用可能な戻り値一覧

returns に定義されています。

```
('pipe',  ('error', 'pipe',)),
```

上記が表しているのは vcl_pipe では return する際に error と pipe を指定できるということです。

変数の一覧

sp_variables に req.url などの変数の一覧とそれぞれがどこのアクションでどのように利用可能か・型名などが以下のように定義されています。

```
('bereq.between_bytes_timeout',    //変数名
        'DURATION',                //変数の型
        ( 'pass', 'miss',),        //変数を読み込む事が可能なアクションリスト
        ( 'pass', 'miss',),        //変数を書きこむ事が可能なアクションリスト
        'struct sess *'            //読み書きする際の関数の引数の Prefix
),
```

アクションの指定で全てのアクションで使える all と vcl_ini,vcl_fini を除いた全てで使える proc があります。

ストレージ変数一覧

stv_variables にストレージの変数が定義されています

VCL での変数の型一覧

vcltypes に使用可能な型が定義されています。それぞれ VCL での型名とそれを C に解釈した際の型がマッピングされています。

この generate.py はその名の通りファイルを生成するもので、以下のファイルを生成します。

```
/lib/libvcl/vcc_token_defs.h
/include/vcl_returns.h
/include/vcl.h
/include/vrt_obj.h
/lib/libvcl/vcc_obj.c
/lib/libvcl/vcc_fixed_token.c
/include/vrt_stv_var.h
```

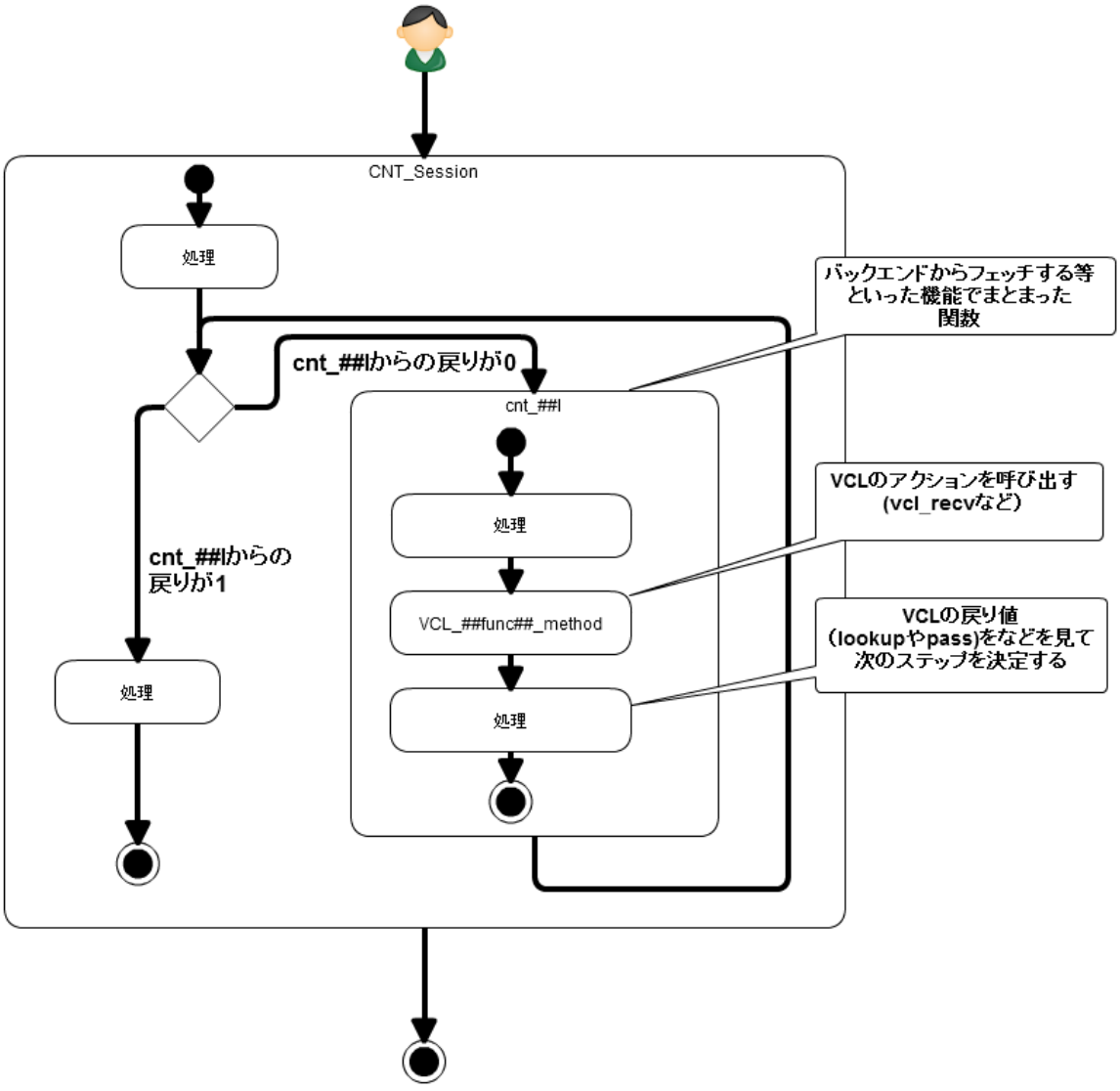
/lib/libvcl/vcc_obj.c

generate.py から生成された VCL で利用可能な変数の一覧で、以下のように定義されています。

```
{ "bereq.between_bytes_timeout", DURATION, 27,    //変数名,型名,変数名の長さ
    "VRT_r_bereq_between_bytes_timeout(sp)",      //変数を読み込む際の関数名
    VCL_MET_PASS | VCL_MET_MISS,                  //変数を読み込むことができるアクション
    "VRT_l_bereq_between_bytes_timeout(sp, ",     //変数を書きこむ際の関数名
    VCL_MET_PASS | VCL_MET_MISS,                  //変数を書きこむことができるアクション
    0,
},
```

/bin/varnishd/cache_center.c

このファイルを見ればたいていのVarnishの動きは把握できます。
単純なインラインCを扱う場合はあまり意識をする必要はありませんが、Varnishをより深く知るためには避けられないファイルです。
例えばvcl_hashはどのタイミングで呼び出されるでしょうか？バックエンドへのフェッチはどのタイミングで？そのような処理が全てまとまっています。
下図を参照してください。



おおまかに言うとVarnishがリクエストを処理する際CNT_Sessionを起点として、機能でまとまったステップを呼んでいき処理していきます。