```
diff -u -d -r -N JS1.5ORG/jsapi.c JS1.5MOD/jsapi.c
--- JS1.5ORG/jsapi.c    2005-10-22 19:19:14.000000000 +0900
+++ JS1.5MOD/jsapi.c    2011-03-04 16:45:58.000000000 +0900
@@ -665,7 +665,9 @@

     if (!js_InitStringGlobals())
         return NULL;
-    rt = (JSRuntime *) malloc(sizeof(JSRuntime));
+    /* rt = (JSRuntime *) SafeMalloc(sizeof(JSRuntime)); 2007-02-26*/
+    rt = (JSRuntime *) JS_PAGE_MALLOC(sizeof(JSRuntime)); // 2007/04/25
+
     if (!rt)
         return NULL;

@@ -752,7 +754,8 @@
         JS_DESTROY_CONDVAR(rt->scopeSharingDone);
 #endif
     js_FinishPropertyTree(rt);
-    free(rt);
+    //SafeFree(rt);/*2007-02-26*/
+   JS_PAGE_FREE(rt); // 2007/04/25
 }

 JS_PUBLIC_API(void)
@@ -1159,14 +1162,14 @@
     /* Initialize the rest of the standard objects and functions. */
     return js_InitArrayClass(cx, obj) &&
            js_InitBooleanClass(cx, obj) &&
-           js_InitMathClass(cx, obj) &&
+           /*js_InitMathClass(cx, obj) && *//*2007-02-26*/
            js_InitNumberClass(cx, obj) &&
            js_InitStringClass(cx, obj) &&
 #if JS_HAS_CALL_OBJECT
            js_InitCallClass(cx, obj) &&
 #endif
 #if JS_HAS_REGEXPS
-           js_InitRegExpClass(cx, obj) &&
+           /* js_InitRegExpClass(cx, obj) &&  no RegExp in ARIB, 2007/03/29*/
 #endif
 #if JS_HAS_SCRIPT_OBJECT
            js_InitScriptClass(cx, obj) &&
@@ -1196,7 +1199,7 @@
     {js_InitArrayClass,             ATOM_OFFSET(Array)},
     {js_InitBooleanClass,           ATOM_OFFSET(Boolean)},
     {js_InitDateClass,              ATOM_OFFSET(Date)},
-    {js_InitMathClass,              ATOM_OFFSET(Math)},
+    /*{js_InitMathClass,              ATOM_OFFSET(Math)}*/ /*2007-02-26*/
     {js_InitNumberClass,            ATOM_OFFSET(Number)},
     {js_InitStringClass,            ATOM_OFFSET(String)},
 #if JS_HAS_CALL_OBJECT
@@ -1461,7 +1464,8 @@
     if (nbytes == 0)
         nbytes = 1;
     cx->runtime->gcMallocBytes += nbytes;
-    p = malloc(nbytes);
+    // p = SafeMalloc(nbytes); /* p = malloc(nbytes); 2007-02-26*/
+    p = JS_PAGE_MALLOC(nbytes); // 2007/04/25
     if (!p)
         JS_ReportOutOfMemory(cx);
     return p;
@@ -1470,7 +1474,8 @@
 JS_realloc(JSContext *cx, void *p, size_t nbytes)
 {
-    p = realloc(p, nbytes);
+    /*p = SafeRealloc(p , nbytes); p = realloc(p, nbytes); 2007-02-26*/
+   p = JS_PAGE_REALLOC(p, nbytes); // 2007/04/04
     if (!p)
         JS_ReportOutOfMemory(cx);
     return p;
@@ -1480,7 +1485,8 @@
 JS_free(JSContext *cx, void *p)
 {
     if (p)
-        free(p);
+        //SafeFree(p);/*free(p); 2007-02-26*/
+       JS_PAGE_FREE(p); // 2007/04/25
 }

 JS_PUBLIC_API(char *)
diff -u -d -r -N JS1.5ORG/jsapi.h JS1.5MOD/jsapi.h
--- JS1.5ORG/jsapi.h    2004-09-02 05:51:38.000000000 +0900
+++ JS1.5MOD/jsapi.h    2011-03-04 16:37:06.000000000 +0900
@@ -43,7 +43,8 @@
  * JavaScript API.
  */
 #include <stddef.h>
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include "jspubtd.h"

 JS_BEGIN_EXTERN_C
@@ -68,7 +69,11 @@
 /* Predicates for type testing. */
 #define JSVAL_IS_OBJECT(v)      (JSVAL_TAG(v) == JSVAL_OBJECT)
 #define JSVAL_IS_NUMBER(v)      (JSVAL_IS_INT(v) || JSVAL_IS_DOUBLE(v))
+#if 0  //change 2009/10/21 for QAC [MISRA Rule 47]
```

```
 #define JSVAL_IS_INT(v)         (((v) & JSVAL_INT) && (v) != JSVAL_VOID)
+#else
+#define JSVAL_IS_INT(v)         (((v) & JSVAL_INT) && ((v) != JSVAL_VOID))
+#endif
 #define JSVAL_IS_DOUBLE(v)      (JSVAL_TAG(v) == JSVAL_DOUBLE)
 #define JSVAL_IS_STRING(v)      (JSVAL_TAG(v) == JSVAL_STRING)
 #define JSVAL_IS_BOOLEAN(v)     (JSVAL_TAG(v) == JSVAL_BOOLEAN)
@@ -96,12 +101,19 @@

 /* Domain limits for the jsval int type. */
 #define JSVAL_INT_BITS          31
-#define JSVAL_INT_POW2(n)       ((jsval)1 << (n))
+#define JSVAL_INT_POW2(n)       (((jsval)1) << (n))   //change 2009/12/16 for QAC [MISRA Rule 47]
 #define JSVAL_INT_MIN           ((jsval)1 - JSVAL_INT_POW2(30))
 #define JSVAL_INT_MAX           (JSVAL_INT_POW2(30) - 1)
+/*20070410*/
+/*On ROSA,we need to convert the type of rval to force them to be compared as Unsigned Integer.*/
+/*else they will be compared as Signed Integer*/
+#ifdef _ROSA
+#define INT_FITS_IN_JSVAL(i)    ((jsuint)((i)+JSVAL_INT_MAX) <= (jsuint)(2*JSVAL_INT_MAX))
+#else
 #define INT_FITS_IN_JSVAL(i)    ((jsuint)((i)+JSVAL_INT_MAX) <= 2*JSVAL_INT_MAX)
+#endif
 #define JSVAL_TO_INT(v)         ((jsint)(v) >> 1)
-#define INT_TO_JSVAL(i)         (((jsval)(i) << 1) | JSVAL_INT)
+#define INT_TO_JSVAL(i)         ((((jsval)(i)) << 1) | JSVAL_INT)  //change 2009/12/16 for QAC [MISRA Rule 47]

 /* Convert between boolean and jsval. */
 #define JSVAL_TO_BOOLEAN(v)     ((JSBool)((v) >> JSVAL_TAGBITS))
diff -u -d -r -N JS1.5ORG/jsarena.c JS1.5MOD/jsarena.c
--- JS1.5ORG/jsarena.c  2004-08-29 03:43:50.000000000 +0900
+++ JS1.5MOD/jsarena.c  2011-03-04 14:04:02.000000000 +0900
@@ -191,7 +191,8 @@

             /* Nothing big enough on the freelist, so we must malloc. */
             JS_RELEASE_LOCK(arena_freelist_lock);
-            b = (JSArena *) malloc(gross);
+            // b = (JSArena *) SafeMalloc(gross);/*2007-02-26*/
+           b = (JSArena *) JS_PAGE_MALLOC(gross); // 2007/04/25
             if (!b)
                 return 0;
             b->next = NULL;
@@ -247,7 +248,8 @@
     extra = HEADER_SIZE(pool);                  /* oversized header holds ap */
     hdrsz = sizeof *a + extra + pool->mask;     /* header and alignment slop */
     gross = hdrsz + aoff;
-    a = (JSArena *) realloc(a, gross);
+    //a = (JSArena *) SafeRealloc(a, gross);/*2007-02-26*/
+   a = (JSArena *) JS_PAGE_REALLOC(a, gross); // 2007/04/04
     if (!a)
         return NULL;
 #ifdef JS_ARENAMETER
@@ -328,7 +330,8 @@
         *ap = a->next;
         JS_CLEAR_ARENA(a);
         JS_COUNT_ARENA(pool,--);
-        free(a);
+            //SafeFree(a);/*2007-02-26*/
+        JS_PAGE_FREE(a); // 2007/04/25
     } while ((a = *ap) != NULL);
     } else {
     /* Insert the whole arena chain at the front of the freelist. */
@@ -423,8 +426,9 @@
     }
     JS_CLEAR_ARENA(a);
     JS_COUNT_ARENA(pool,--);
-    free(a);
-}
+    //SafeFree(a);/*2007-02-26*/
+   JS_PAGE_FREE(a); // 2007/04/25
+}

 JS_PUBLIC_API(void)
 JS_FreeArenaPool(JSArenaPool *pool)
@@ -442,7 +446,8 @@
     JSArenaStats *stats, **statsp;

     if (pool->stats.name)
-        free(pool->stats.name);
+            //SafeFree(pool->stats.name);/*2007-02-26*/
+           JS_PAGE_FREE(pool->stats.name); // 2007/04/25
     for (statsp = &arena_stats_list; (stats = *statsp) != 0;
          statsp = &stats->next) {
         if (stats == &pool->stats) {
@@ -465,7 +470,8 @@
     JS_RELEASE_LOCK(arena_freelist_lock);
     for (; a; a = next) {
         next = a->next;
-        free(a);
+        //SafeFree(a);/*2007-02-26*/
+       JS_PAGE_FREE(a); // 2007/04/25
     }
 }

@@ -523,7 +529,8 @@
 }

 #include <math.h>
```

```
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"

 JS_PUBLIC_API(void)
 JS_DumpArenaStats(FILE *fp)
diff -u -d -r -N JS1.5ORG/jsarena.h JS1.5MOD/jsarena.h
--- JS1.5ORG/jsarena.h  2003-11-15 09:10:56.000000000 +0900
+++ JS1.5MOD/jsarena.h  2011-03-04 15:25:06.000000000 +0900
@@ -196,7 +196,7 @@
         if ((pool)->current == (a)) (pool)->current = &(pool)->first;         ¥
         *(pnext) = (a)->next;                                                 ¥
         JS_CLEAR_ARENA(a);                                                    ¥
-        free(a);                                                              ¥
+        JS_PAGE_FREE(a);/* 2007/04/25 SafeFree(a); 2007-02-26*/¥
         (a) = NULL;                                                           ¥
     JS_END_MACRO

@@ -267,7 +267,8 @@

 #ifdef JS_ARENAMETER

-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"

 extern JS_PUBLIC_API(void)
 JS_ArenaCountAllocation(JSArenaPool *pool, size_t nb);
diff -u -d -r -N JS1.5ORG/jsarray.c JS1.5MOD/jsarray.c
--- JS1.5ORG/jsarray.c  2004-06-03 06:10:00.000000000 +0900
+++ JS1.5MOD/jsarray.c  2011-03-04 14:06:52.000000000 +0900
@@ -359,16 +359,19 @@
         growth = (1 + 3 + 1) * sizeof(jschar);
         if (!chars) {
             nchars = 0;
-            chars = (jschar *) malloc(growth);
+            //chars = (jschar *) SafeMalloc(growth);/*2007-02-26*/
+           chars = (jschar *) JS_PAGE_MALLOC(growth); // 2007/04/25
             if (!chars)
                 goto done;
         } else {
             MAKE_SHARP(he);
             nchars = js_strlen(chars);
             chars = (jschar *)
-                realloc((ochars = chars), nchars * sizeof(jschar) + growth);
+                //SafeRealloc((ochars = chars), nchars * sizeof(jschar) + growth);/*2007-02-26*/
+               JS_PAGE_REALLOC((ochars = chars), nchars * sizeof(jschar) + growth);// 2007/04/04
             if (!chars) {
-                free(ochars);
+                //SafeFree(ochars);/*2007-02-26*/
+               JS_PAGE_FREE(ochars); // 2007/04/25
                 goto done;
             }
         }
@@ -430,13 +433,16 @@
                     JSSTRING_LENGTH(str) +
                     3 + 1) * sizeof(jschar);
         if (!chars) {
-            chars = (jschar *) malloc(growth);
+            // chars = (jschar *) SafeMalloc(growth);/*2007-02-26*/
+           chars = (jschar *) JS_PAGE_MALLOC(growth); // 2007/04/25
             if (!chars)
                 goto done;
         } else {
-            chars = (jschar *) realloc((ochars = chars), growth);
+            //chars = (jschar *) SafeRealloc((ochars = chars), growth);/*2007-02-26*/
+           chars = (jschar *) JS_PAGE_REALLOC((ochars = chars), growth);// 2007/04/04
             if (!chars) {
-                free(ochars);
+                //SafeFree(ochars);/*2007-02-26*/
+               JS_PAGE_FREE(ochars); // 2007/04/25
                 goto done;
             }
         }
@@ -467,7 +473,8 @@
     js_LeaveSharpObject(cx, NULL);
     if (!ok) {
         if (chars)
-            free(chars);
+            //SafeFree(chars);/*2007-02-26*/
+           JS_PAGE_FREE(chars);// 2007/04/25
         return ok;
     }

@@ -479,7 +486,8 @@
     chars[nchars] = 0;
     str = js_NewString(cx, chars, nchars, 0);
     if (!str) {
-        free(chars);
+        //SafeFree(chars);/*2007-02-26*/
+       JS_PAGE_FREE(chars);// 2007/04/25
         return JS_FALSE;
     }
     *rval = STRING_TO_JSVAL(str);
@@ -710,7 +718,8 @@
     HSortArgs hsa;
     size_t i;
```

```
-    pivot = malloc(elsize);
+    // pivot = SafeMalloc(elsize);/*2007-02-26*/
+   pivot = JS_PAGE_MALLOC(elsize); // 2007/04/25
     if (!pivot)
         return JS_FALSE;
     hsa.vec = vec;
@@ -725,7 +734,8 @@
     while (nel > 2)
         HeapSortHelper(JS_FALSE, &hsa, 1, --nel);

-    free(pivot);
+    //SafeFree(pivot);/*2007-02-26*/
+   JS_PAGE_FREE(pivot); // 2007/04/25
     return JS_TRUE;
 }

@@ -745,6 +755,19 @@
     jsval fval, argv[2], rval;
     JSBool ok;

+// Bug-list HDTV-BML No00240 ADD-START
+    // undefinedを含んだ比較の場合
+    if( (av == JSVAL_VOID) && (bv == JSVAL_VOID) ) {
+        return 0;
+    }
+    else if( av == JSVAL_VOID ) {
+        return 1;
+    }
+    else if( bv == JSVAL_VOID ) {
+        return -1;
+    }
+// Bug-list HDTV-BML No00240 ADD-END
+
     fval = ca->fval;
     if (fval == JSVAL_NULL) {
         JSString *astr, *bstr;
diff -u -d -r -N JS1.5ORG/jsatom.c JS1.5MOD/jsatom.c
--- JS1.5ORG/jsatom.c   2004-06-22 02:57:10.000000000 +0900
+++ JS1.5MOD/jsatom.c   2011-03-04 14:07:46.000000000 +0900
@@ -135,6 +135,9 @@
 const char js_ExecutionContext_str[] = "ExecutionContext";
 const char js_current_str[]          = "current";
 #endif
+// Bug-list HDTV-BML No00237 START
+const char js_hostobject_str[]       = "hostobject";
+// Bug-list HDTV-BML No00237 END

 #define HASH_OBJECT(o)  ((JSHashNumber)(o) >> JSVAL_TAGBITS)
 #define HASH_INT(i)     ((JSHashNumber)(i))
@@ -197,13 +200,15 @@
 void * JS_DLL_CALLBACK
 js_alloc_table_space(void *priv, size_t size)
 {
-    return malloc(size);
+    //return SafeMalloc(size);/*2007-02-26*/
+   return JS_PAGE_MALLOC(size); // 2007/04/25
 }

 void JS_DLL_CALLBACK
 js_free_table_space(void *priv, void *item)
 {
-    free(item);
+    //SafeFree(item);/*2007-02-26*/
+   JS_PAGE_FREE(item); // 2007/04/25
 }

 JS_STATIC_DLL_CALLBACK(JSHashEntry *)
@@ -212,7 +217,8 @@
     JSAtomState *state = (JSAtomState *) priv;
     JSAtom *atom;

-    atom = (JSAtom *) malloc(sizeof(JSAtom));
+    //atom = (JSAtom *) SafeMalloc(sizeof(JSAtom));/*2007-02-26*/
+   atom = (JSAtom *) JS_PAGE_MALLOC(sizeof(JSAtom)); // 2007/04/25
     if (!atom)
         return NULL;
 #ifdef JS_THREADSAFE
@@ -233,7 +239,8 @@
 #ifdef JS_THREADSAFE
     ((JSAtomState *)priv)->tablegen++;
 #endif
-    free(he);
+    //SafeFree(he);/*2007-02-26*/
+   JS_PAGE_FREE(he);// 2007/04/25
 }

 static JSHashAllocOps atom_alloc_ops = {
@@ -331,6 +338,9 @@
     FROB(ExecutionContextAtom,    js_ExecutionContext_str);
     FROB(currentAtom,             js_current_str);
 #endif
+// Bug-list HDTV-BML No00237 START
+    FROB(hostobjectAtom,          js_hostobject_str);
+// Bug-list HDTV-BML No00237 END

 #undef FROB

diff -u -d -r -N JS1.5ORG/jsatom.h JS1.5MOD/jsatom.h
```

```
--- JS1.5ORG/jsatom.h   2004-02-11 16:21:58.000000000 +0900
+++ JS1.5MOD/jsatom.h   2011-02-22 09:02:28.000000000 +0900
@@ -241,6 +241,9 @@
     JSAtom              *ExecutionContextAtom;
     JSAtom              *currentAtom;
 #endif
+// Bug-list HDTV-BML No00237 START
+    JSAtom              *hostobjectAtom;
+// Bug-list HDTV-BML No00237 END
 };

 /* Well-known predefined strings and their atoms. */
@@ -291,6 +294,9 @@
 extern const char   js_ExecutionContext_str[];
 extern const char   js_current_str[];
 #endif
+// Bug-list HDTV-BML No00237 START
+extern const char js_hostobject_str[];
+// Bug-list HDTV-BML No00237 END

 /*
  * Initialize atom state.  Return true on success, false with an out of
diff -u -d -r -N JS1.5ORG/jscntxt.c JS1.5MOD/jscntxt.c
--- JS1.5ORG/jscntxt.c  2004-08-20 02:57:36.000000000 +0900
+++ JS1.5MOD/jscntxt.c  2011-03-04 14:08:42.000000000 +0900
@@ -69,7 +69,8 @@
     JSContext *cx;
     JSBool ok, first;

-    cx = (JSContext *) malloc(sizeof *cx);
+    // cx = (JSContext *) SafeMalloc(sizeof *cx);/*2007-02-26*/
+   cx = (JSContext *) JS_PAGE_MALLOC(sizeof *cx); // 2007/04/25
     if (!cx)
         return NULL;
     memset(cx, 0, sizeof *cx);
@@ -261,7 +262,8 @@
     JS_FinishArenaPool(&cx->stackPool);
     JS_FinishArenaPool(&cx->tempPool);
     if (cx->lastMessage)
-        free(cx->lastMessage);
+        //SafeFree(cx->lastMessage);/*2007-02-26*/
+       JS_PAGE_FREE(cx->lastMessage);// 2007/04/25

     /* Remove any argument formatters. */
     map = cx->argumentFormatMap;
@@ -287,7 +289,8 @@
     }

     /* Finally, free cx itself. */
-    free(cx);
+    //SafeFree(cx);/*2007-02-26*/
+   JS_PAGE_FREE(cx);// 2007/04/25
 }

 JSBool
@@ -753,7 +756,8 @@
     }

     ReportError(cx, last, &report);
-    free(last);
+    //SafeFree(last);/*2007-02-26*/
+   JS_PAGE_FREE(last);// 2007/04/25
     return warning;
 }

@@ -976,7 +980,8 @@
         return;

     if (cx->lastMessage)
-        free(cx->lastMessage);
+        //SafeFree(cx->lastMessage);/*2007-02-26*/
+       JS_PAGE_FREE(cx->lastMessage);// 2007/04/25
     cx->lastMessage = JS_strdup(cx, message);
     if (!cx->lastMessage)
         return;
diff -u -d -r -N JS1.5ORG/jsconfig.h JS1.5MOD/jsconfig.h
--- JS1.5ORG/jsconfig.h 2004-07-16 04:21:34.000000000 +0900
+++ JS1.5MOD/jsconfig.h 2011-03-04 15:39:06.000000000 +0900
@@ -443,8 +443,8 @@
 #define JS_HAS_SWITCH_STATEMENT 1       /* has switch (v) {case c: ...} */
 #define JS_HAS_SOME_PERL_FUN    1       /* has array.join/reverse/sort */
 #define JS_HAS_MORE_PERL_FUN    1       /* has array.push, str.substr, etc */
-#define JS_HAS_STR_HTML_HELPERS 1       /* has str.anchor, str.bold, etc. */
-#define JS_HAS_PERL_SUBSTR      1       /* has str.substr */
+#define JS_HAS_STR_HTML_HELPERS 0       /* has str.anchor, str.bold, etc. */ // from 0 to 1, 2007/03/30
+#define JS_HAS_PERL_SUBSTR      0       /* has str.substr */// from 0 to 1, 2007/03/30
 #define JS_HAS_VALUEOF_HINT     1       /* valueOf(hint) where hint is typeof */
 #define JS_HAS_LEXICAL_CLOSURE  1       /* nested functions, lexically closed */
 #define JS_HAS_APPLY_FUNCTION   1       /* has apply(fun, arg1, ... argN) */
diff -u -d -r -N JS1.5ORG/jscpucfg.h JS1.5MOD/jscpucfg.h
--- JS1.5ORG/jscpucfg.h 2004-04-24 04:07:40.000000000 +0900
+++ JS1.5MOD/jscpucfg.h 2011-03-04 15:39:20.000000000 +0900
@@ -191,6 +191,52 @@
 #error "static version for Mac and Windows platforms is being used."
 #error "Something's probably wrong with paths/headers/dependencies/Makefiles."

+#elif defined (_ROSA)/*2007-02-26*/
+#define IS_LITTLE_ENDIAN 1
```

```
+#undef  IS_BIG_ENDIAN
+
+#define JS_BYTES_PER_BYTE   1L
+#define JS_BYTES_PER_SHORT  2L
+#define JS_BYTES_PER_INT    4L
+#define JS_BYTES_PER_INT64  8L
+#define JS_BYTES_PER_LONG   4L
+#define JS_BYTES_PER_FLOAT  4L
+#define JS_BYTES_PER_DOUBLE 8L
+#define JS_BYTES_PER_WORD   4L
+#define JS_BYTES_PER_DWORD  8L
+
+#define JS_BITS_PER_BYTE    8L
+#define JS_BITS_PER_SHORT   16L
+#define JS_BITS_PER_INT     32L
+#define JS_BITS_PER_INT64   64L
+#define JS_BITS_PER_LONG    32L
+#define JS_BITS_PER_FLOAT   32L
+#define JS_BITS_PER_DOUBLE  64L
+#define JS_BITS_PER_WORD    32L
+
+#define JS_BITS_PER_BYTE_LOG2   3L
+#define JS_BITS_PER_SHORT_LOG2  4L
+#define JS_BITS_PER_INT_LOG2    5L
+#define JS_BITS_PER_INT64_LOG2  6L
+#define JS_BITS_PER_LONG_LOG2   5L
+#define JS_BITS_PER_FLOAT_LOG2  5L
+#define JS_BITS_PER_DOUBLE_LOG2 6L
+#define JS_BITS_PER_WORD_LOG2   5L
+
+#define JS_ALIGN_OF_SHORT   2L
+#define JS_ALIGN_OF_INT     4L
+#define JS_ALIGN_OF_LONG    4L
+#define JS_ALIGN_OF_INT64   4L
+#define JS_ALIGN_OF_FLOAT   4L
+#define JS_ALIGN_OF_DOUBLE  4L
+#define JS_ALIGN_OF_POINTER 4L
+#define JS_ALIGN_OF_WORD    4L
+
+#define JS_BYTES_PER_WORD_LOG2   2L
+#define JS_BYTES_PER_DWORD_LOG2  3L
+#define JS_WORDS_PER_DWORD_LOG2  1L
+
+#define JS_STACK_GROWTH_DIRECTION (-1)
 #else

 #error "Must define one of XP_BEOS, XP_MAC, XP_OS2, XP_WIN, or XP_UNIX"
diff -u -d -r -N JS1.5ORG/jsdate.c JS1.5MOD/jsdate.c
--- JS1.5ORG/jsdate.c   2004-06-16 01:38:42.000000000 +0900
+++ JS1.5MOD/jsdate.c   2011-03-04 15:31:24.000000000 +0900
@@ -52,7 +52,7 @@

 #include "jsstddef.h"
 #include <ctype.h>
-#include <locale.h>
+/*#include <locale.h> 2007-02-26*/
 #include <math.h>
 #include <stdlib.h>
 #include <string.h>
@@ -69,6 +69,9 @@
 #include "jsobj.h"
 #include "jsstr.h"

+#ifdef _ROSA   /*need call ast_ifsync_GetTime() 20070405*/
+#include <ROSA.h>
+#endif
 /*
  * The JS 'Date' object is patterned after the Java 'Date' object.
  * Here is an script:
@@ -317,7 +320,7 @@
 static jsdouble
 DaylightSavingTA(jsdouble t)
 {
-    volatile int64 PR_t;
+    int64 PR_t;
     int64 ms2us;
     int64 offset;
     jsdouble result;
@@ -404,7 +407,11 @@

     leap = (DaysInYear((jsint) year) == 366);

+#ifdef _ROSA /*2007/04/17*/
+yearday = DayFromYear(year);
+#else
     yearday = floor(TimeFromYear(year) / msPerDay);
+#endif
     monthday = DayFromMonth(month, leap);

     result = yearday
@@ -1433,14 +1440,23 @@
    /* Avoid dependence on PRMJ_FormatTimeUSEnglish, because it
     * requires a PRMJTime... which only has 16-bit years.  Sub-ECMA.
     */
-   JS_snprintf(buf, sizeof buf, "%s, %.2d %s %.4d %.2d:%.2d:%.2d GMT",
-           days[WeekDay(temp)],
-           DateFromTime(temp),
-           months[MonthFromTime(temp)],
```

```
-           YearFromTime(temp),
-           HourFromTime(temp),
-           MinFromTime(temp),
-           SecFromTime(temp));
+// Bug-list HDTV-BML #00297 CHG-START
+// JS_snprintf(buf, sizeof buf, "%s, %.2d %s %.4d %.2d:%.2d:%.2d GMT",
+//          days[WeekDay(temp)],
+//          DateFromTime(temp),
+//          months[MonthFromTime(temp)],
+//          YearFromTime(temp),
+//          HourFromTime(temp),
+//          MinFromTime(temp),
+//          SecFromTime(temp));
+     JS_snprintf(buf, sizeof buf, "%.4d-%.2d-%.2dT%.2d:%.2d:%.2d",
+             YearFromTime(temp),
+             (MonthFromTime(temp)+1),
+             DateFromTime(temp),
+             HourFromTime(temp),
+             MinFromTime(temp),
+             SecFromTime(temp));
+// Bug-list HDTV-BML #00297 CHG-END
     }
     str = JS_NewStringCopyZ(cx, buf);
     if (!str)
@@ -1495,7 +1511,7 @@
 }

 typedef enum formatspec {
-    FORMATSPEC_FULL, FORMATSPEC_DATE, FORMATSPEC_TIME
+    FORMATSPEC_FULL, FORMATSPEC_DATE, FORMATSPEC_TIME,FORMATSPEC_ARIB
 } formatspec;

 /* helper function */
@@ -1581,6 +1597,27 @@
                         offset,
                         usetz ? " " : "",
                         usetz ? tzbuf : "");
+          break;
+         case FORMATSPEC_ARIB:
+          /*copy from ARIB STD-B24 Version5.0-E1*/
+          /* 20070330 */
+          /*  Result of Date.prototype.toString()
+          Must be in the format of "DateThoursminutesseconds."
+          Date must be YYYY-MM-DD. (Ex. 1999-01-01)
+          Hours, minutes and seconds must be hh:mm:ss. (Ex.: 23:01:34)
+          'T' (character code 0x54) must be used as a delimiter between the date and the hours, minutes and
+          seconds. (Ex.: 1999-01-01T23:01:34)
+          If the result is a negative value, the low-order four digits are used and the sign (d.c. or a.c.) is
+          ignored.
+          */
+           JS_snprintf(buf, sizeof buf,
+                       "%.4d-%.2d-%.2dT%.2d:%.2d:%.2d",
+                       YearFromTime(local),
+                      (MonthFromTime(local)+1),
+                      DateFromTime(local),
+                       HourFromTime(local),
+                       MinFromTime(local),
+                       SecFromTime(local));
           break;
         case FORMATSPEC_DATE:
           /* Tue Oct 31 2000 */
@@ -1635,8 +1672,10 @@

     /* If it failed, default to toString. */
     if (result_len == 0)
-        return date_format(cx, *date, FORMATSPEC_FULL, rval);
-
+// Bug-list HDTV-BML #00297 CHG-START
+         return date_format(cx, *date, FORMATSPEC_ARIB, rval);
+//      return date_format(cx, *date, FORMATSPEC_FULL, rval);
+// Bug-list HDTV-BML #00297 CHG-END
         /* Hacked check against undesired 2-digit year 00/00/00 form. */
         if (buf[result_len - 3] == '/' &&
             isdigit(buf[result_len - 2]) && isdigit(buf[result_len - 1])) {
@@ -1747,7 +1786,8 @@

     str = JS_NewString(cx, bytes, strlen(bytes));
     if (!str) {
-    free(bytes);
+        //SafeFree(bytes);/*2007-02-26*/
+        JS_PAGE_FREE(bytes);// 2007/04/25
     return JS_FALSE;
     }
     *rval = STRING_TO_JSVAL(str);
@@ -1761,8 +1801,9 @@
 {
     jsdouble *date = date_getProlog(cx, obj, argv);
     if (!date)
-    return JS_FALSE;
-    return date_format(cx, *date, FORMATSPEC_FULL, rval);
+        return JS_FALSE;
+    //return date_format(cx, *date, FORMATSPEC_FULL, rval);FORMATSPEC_ARIB
+    return date_format(cx, *date, FORMATSPEC_ARIB, rval);
 }

 #if JS_HAS_VALUEOF_HINT
@@ -1890,7 +1931,10 @@
     JSLL_DIV(ms, us, us2ms);
```

```
    JSLL_L2D(msec_time, ms);

-   return date_format(cx, msec_time, FORMATSPEC_FULL, rval);
+//Bug-list HDTV-BML No00296 START
+// return date_format(cx, msec_time, FORMATSPEC_FULL, rval);
+   return date_format(cx, msec_time, FORMATSPEC_ARIB, rval);
+//Bug-list HDTV-BML No00296 END
     }

     /* Date called as constructor */
@@ -1901,13 +1945,41 @@
    date = date_constructor(cx, obj);
    if (!date)
        return JS_FALSE;
+#if 0 /*TEMP fixed 20070403*/
+   {
+       jsdouble array[MAXARGS];
+       jsdouble day;
+       jsdouble msec_time;
+       AST_TIME_TBL time;
+       ER ercd = E_OK;

+       ercd = ast_ifsync_GetTime(&time, 1);/*mode 1?*/
+       if(E_OK != ercd)
+       {
+           //Trace(TRACE_ERR, "[EBML]ast_ifsync_GetTime err @new Date()");
+           /*set default time*/
+       }
+       array[0] = (jsdouble)time.year;
+       array[1] = (jsdouble)(time.month-1);
+       array[2] = (jsdouble)time.day;
+       array[3] = (jsdouble)time.hour;
+       array[4] = (jsdouble)time.minute;
+       array[5] = (jsdouble)time.second;
+       array[6] = (jsdouble)time.msec;
+       day = MakeDay(array[0], array[1], array[2]);
+       msec_time = MakeTime(array[3], array[4], array[5], array[6]);
+       msec_time = MakeDate(day, msec_time);
+       msec_time = UTC(msec_time);
+       *date = TIMECLIP(msec_time);
+   }
+#else
    us = PRMJ_Now();
    JSLL_UI2L(us2ms, PRMJ_USEC_PER_MSEC);
    JSLL_DIV(ms, us, us2ms);
    JSLL_L2D(msec_time, ms);

    *date = msec_time;
+#endif /*20070403 end*/
     } else if (argc == 1) {
    if (!JSVAL_IS_STRING(argv[0])) {
        /* the argument is a millisecond number */
@@ -2236,3 +2308,58 @@
         return 0;
     return (*date);
 }
+
+#ifdef _ROSA /*20070409*/
+JSInt64 rosa_prmj_now()
+{
+       jsdouble array[MAXARGS];
+       jsdouble day;
+       JSInt64 intday;
+       JSInt64 msec_time;
+       AST_TIME_TBL time;
+       JSInt64 ms_perday;
+       JSInt64 usec_persec;
+       JSInt64 diff_msec, diff;
+       ER ercd = E_OK;
+
+       ercd = ast_ifsync_GetTime(&time, 1);/*mode 1?*/
+       if(E_OK != ercd)
+       {
+           //Trace(TRACE_ERR, "[EBML]ast_ifsync_GetTime err @new Date()");
+           /*set default time*/
+       }
+       array[0] = (jsdouble)time.year;
+       array[1] = (jsdouble)(time.month-1);
+       array[2] = (jsdouble)time.day;
+       array[3] = (jsdouble)time.hour;
+       array[4] = (jsdouble)time.minute;
+       array[5] = (jsdouble)time.second;
+       array[6] = (jsdouble)time.msec;
+
+       day = MakeDay(array[0], array[1], array[2]);
+       JSLL_D2L(intday,day);
+       JSLL_D2L(msec_time,MakeTime(array[3], array[4], array[5], array[6]));
+
+       //msec_time = MakeDate(day, msec_time);
+
+       JSLL_D2L(ms_perday, msPerDay);
+       JSLL_MUL(intday,intday, ms_perday);
+
+       JSLL_ADD(msec_time,intday,msec_time);
+
+       JSLL_I2L(usec_persec, PRMJ_USEC_PER_MSEC);
+
+       JSLL_MUL(msec_time,msec_time,usec_persec);
```

```
+       //msec_time = UTC(msec_time);
+       JSLL_I2L(diff_msec, (PRMJ_LocalGMTDifference())*PRMJ_USEC_PER_MSEC);
+
+       JSLL_MUL(diff,diff_msec,usec_persec);
+
+       JSLL_ADD(msec_time, msec_time, diff);
+       //ret = (JSInt64)msec_time * PRMJ_USEC_PER_MSEC;
+
+       return msec_time;
+       //*date = TIMECLIP(msec_time);
+}
+#endif
+
diff -u -d -r -N JS1.5ORG/jsdhash.c JS1.5MOD/jsdhash.c
--- JS1.5ORG/jsdhash.c  2004-07-15 07:14:32.000000000 +0900
+++ JS1.5MOD/jsdhash.c  2011-03-04 14:56:36.000000000 +0900
@@ -40,7 +40,8 @@
 /*
  * Double hashing implementation.
  */
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <stdlib.h>
 #include <string.h>
 #include "jsbit.h"
@@ -59,13 +60,15 @@
 JS_PUBLIC_API(void *)
 JS_DHashAllocTable(JSDHashTable *table, uint32 nbytes)
 {
-    return malloc(nbytes);
+    // return SafeMalloc(nbytes);/*2007-02-26*/
+   return JS_PAGE_MALLOC(nbytes); // 2007/04/25
 }

 JS_PUBLIC_API(void)
 JS_DHashFreeTable(JSDHashTable *table, void *ptr)
 {
-    free(ptr);
+    //SafeFree(ptr);/*2007-02-26*/
+   JS_PAGE_FREE(ptr);// 2007/04/25
 }

 JS_PUBLIC_API(JSDHashNumber)
@@ -135,7 +138,8 @@
 {
     const JSDHashEntryStub *stub = (const JSDHashEntryStub *)entry;

-    free((void *) stub->key);
+    //SafeFree((void *) stub->key);/*2007-02-26*/
+   JS_PAGE_FREE((void *) stub->key);// 2007/04/25
     memset(entry, 0, table->entrySize);
 }

@@ -168,11 +172,13 @@
 {
     JSDHashTable *table;

-    table = (JSDHashTable *) malloc(sizeof *table);
+    //table = (JSDHashTable *) SafeMalloc(sizeof *table);/*2007-02-26*/
+   table = (JSDHashTable *) JS_PAGE_MALLOC(sizeof *table); // 2007/04/25
     if (!table)
         return NULL;
     if (!JS_DHashTableInit(table, ops, data, entrySize, capacity)) {
-        free(table);
+        //SafeFree(table);/*2007-02-26*/
+       JS_PAGE_FREE(table);// 2007/04/25
         return NULL;
     }
     return table;
@@ -182,7 +188,8 @@
 JS_DHashTableDestroy(JSDHashTable *table)
 {
     JS_DHashTableFinish(table);
-    free(table);
+    //SafeFree(table);/*2007-02-26*/
+   JS_PAGE_FREE(table);// 2007/04/25
 }

 JS_PUBLIC_API(JSBool)
diff -u -d -r -N JS1.5ORG/jsdhash.h JS1.5MOD/jsdhash.h
--- JS1.5ORG/jsdhash.h  2004-07-23 05:26:48.000000000 +0900
+++ JS1.5MOD/jsdhash.h  2011-03-04 15:25:30.000000000 +0900
@@ -568,7 +568,8 @@
 JS_DHashTableEnumerate(JSDHashTable *table, JSDHashEnumerator etor, void *arg);

 #ifdef JS_DHASHMETER
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"

 extern JS_PUBLIC_API(void)
 JS_DHashTableDumpMeter(JSDHashTable *table, JSDHashEnumerator dump, FILE *fp);
diff -u -d -r -N JS1.5ORG/jsdtoa.c JS1.5MOD/jsdtoa.c
--- JS1.5ORG/jsdtoa.c   2004-04-04 07:11:10.000000000 +0900
+++ JS1.5MOD/jsdtoa.c   2011-03-04 16:30:26.000000000 +0900
@@ -194,10 +194,14 @@
 #include "stdlib.h"
```

```
 #include "string.h"

+#define MALLOC JS_PAGE_MALLOC // 2007/04/25
+
+#if 0
 #ifdef MALLOC
 extern void *MALLOC(size_t);
 #else
-#define MALLOC malloc
+#define MALLOC SafeMalloc/*2007-02-26*/
+#endif
 #endif

 #define Omit_Private_Memory
@@ -1219,7 +1223,8 @@
         Bigint **listp = &freelist[count];
         while ((temp = *listp) != NULL) {
             *listp = temp->next;
-            free(temp);
+            //SafeFree(temp);/*2007-02-26*/
+           JS_PAGE_FREE(temp);// 2007/04/25
         }
         freelist[count] = NULL;
     }
@@ -1228,7 +1233,8 @@
     while (p5s) {
         temp = p5s;
         p5s = p5s->next;
-        free(temp);
+        //SafeFree(temp);/*2007-02-26*/
+       JS_PAGE_FREE(temp);// 2007/04/25
     }
 }

@@ -2934,7 +2940,8 @@

     JS_ASSERT(base >= 2 && base <= 36);

-    buffer = (char*) malloc(DTOBASESTR_BUFFER_SIZE);
+    //buffer = (char*) SafeMalloc(DTOBASESTR_BUFFER_SIZE);/*2007-02-26*/
+   buffer = (char*) JS_PAGE_MALLOC(DTOBASESTR_BUFFER_SIZE);// 2007/04/25
     if (buffer) {
         p = buffer;
         if (d < 0.0
diff -u -d -r -N JS1.5ORG/jsexn.c JS1.5MOD/jsexn.c
--- JS1.5ORG/jsexn.c    2004-09-21 10:18:00.000000000 +0900
+++ JS1.5MOD/jsexn.c    2011-03-04 14:58:00.000000000 +0900
@@ -512,7 +512,8 @@
              * don't use JS_realloc here; simply let the oversized allocation
              * be owned by the string in that rare case.
              */
-            void *shrunk = realloc(stackbuf, (stacklen+1) * sizeof(jschar));
+            //void *shrunk = SafeRealloc(stackbuf, (stacklen+1) * sizeof(jschar));/*2007-02-26*/
+           void *shrunk = JS_PAGE_REALLOC(stackbuf, (stacklen+1) * sizeof(jschar));// 2007/04/04
             if (shrunk)
                 stackbuf = shrunk;
         }
diff -u -d -r -N JS1.5ORG/jsfile.c JS1.5MOD/jsfile.c
--- JS1.5ORG/jsfile.c   2003-11-15 09:10:56.000000000 +0900
+++ JS1.5MOD/jsfile.c   2011-03-04 16:29:44.000000000 +0900
@@ -124,7 +124,7 @@
 typedef enum JSFileErrNum {
 #define MSG_DEF(name, number, count, exception, format) ¥
     name = number,
-#include "jsfile.msg"
+//#include "jsfile.msg"
 #undef MSG_DEF
     JSFileErr_Limit
 #undef MSGDEF
@@ -140,7 +140,7 @@
 #define MSG_DEF(name, number, count, exception, format) ¥
     { NULL, count } ,
 #endif
-#include "jsfile.msg"
+//#include "jsfile.msg"
 #undef MSG_DEF
 };

diff -u -d -r -N JS1.5ORG/jsgc.c JS1.5MOD/jsgc.c
--- JS1.5ORG/jsgc.c 2004-08-20 02:57:36.000000000 +0900
+++ JS1.5MOD/jsgc.c 2011-03-04 16:29:56.000000000 +0900
@@ -699,8 +699,8 @@

 #ifdef GC_MARK_DEBUG

-#include <stdio.h>
-#include <stdlib.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include "jsprf.h"

 JS_FRIEND_DATA(FILE *) js_DumpGCHeap;
@@ -800,7 +800,8 @@
         break;
     }
     fprintf(fp, " via %s¥n", path);
-    free(path);
+    //SafeFree(path);/*2007-02-26*/
```

```
+   JS_PAGE_FREE(path);// 2007/04/25
 }

 #endif /* !GC_MARK_DEBUG */
diff -u -d -r -N JS1.5ORG/jshash.c JS1.5MOD/jshash.c
--- JS1.5ORG/jshash.c   2004-04-13 10:25:16.000000000 +0900
+++ JS1.5MOD/jshash.c   2011-03-04 15:00:04.000000000 +0900
@@ -67,26 +67,30 @@
 static void *
 DefaultAllocTable(void *pool, size_t size)
 {
-    return malloc(size);
+    // return SafeMalloc(size);/*2007-02-26*/
+   return JS_PAGE_MALLOC(size);// 2007/04/25
 }

 static void
 DefaultFreeTable(void *pool, void *item)
 {
-    free(item);
+    //SafeFree(item);/*2007-02-26*/
+   JS_PAGE_FREE(item);// 2007/04/25
 }

 static JSHashEntry *
 DefaultAllocEntry(void *pool, const void *key)
 {
-    return (JSHashEntry*) malloc(sizeof(JSHashEntry));
+    //return (JSHashEntry*) SafeMalloc(sizeof(JSHashEntry));/*2007-02-26*/
+   return (JSHashEntry*) JS_PAGE_MALLOC(sizeof(JSHashEntry));// 2007/04/25
 }

 static void
 DefaultFreeEntry(void *pool, JSHashEntry *he, uintN flag)
 {
     if (flag == HT_FREE_ENTRY)
-        free(he);
+        //SafeFree(he);/*2007-02-26*/
+       JS_PAGE_FREE(he);// 2007/04/25
 }

 static JSHashAllocOps defaultHashAllocOps = {
@@ -386,7 +390,8 @@

 #ifdef HASHMETER
 #include <math.h>
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"

 JS_PUBLIC_API(void)
 JS_HashTableDumpMeter(JSHashTable *ht, JSHashEnumerator dump, FILE *fp)
diff -u -d -r -N JS1.5ORG/jshash.h JS1.5MOD/jshash.h
--- JS1.5ORG/jshash.h   2003-11-15 09:10:56.000000000 +0900
+++ JS1.5MOD/jshash.h   2011-03-04 16:38:56.000000000 +0900
@@ -43,7 +43,8 @@
  * API to portable hash table code.
  */
 #include <stddef.h>
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include "jstypes.h"
 #include "jscompat.h"

diff -u -d -r -N JS1.5ORG/jsinterp.c JS1.5MOD/jsinterp.c
--- JS1.5ORG/jsinterp.c 2004-09-24 11:16:48.000000000 +0900
+++ JS1.5MOD/jsinterp.c 2011-03-04 16:31:54.000000000 +0900
@@ -46,7 +46,8 @@
  * JavaScript bytecode interpreter.
  */
 #include "jsstddef.h"
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <string.h>
 #include <math.h>
 #include "jstypes.h"
@@ -92,6 +93,10 @@
 #define ASSERT_CACHE_IS_EMPTY(cache) ((void)0)
 #endif

+//Bug-list HDTV-BML No00266 START
+extern JSBool CheckObjGreg(JSContext* cx, JSObject* obj);
+//Bug-list HDTV-BML No00266 END
+
 void
 js_FlushPropertyCache(JSContext *cx)
 {
@@ -671,26 +676,30 @@
 static void *
 AllocCallTable(void *pool, size_t size)
 {
-    return malloc(size);
+    // return SafeMalloc(size);/*2007-02-26*/
+   return JS_PAGE_MALLOC(size);// 2007/04/25
 }
```

```
 static void
 FreeCallTable(void *pool, void *item)
 {
-    free(item);
+    //SafeFree(item);/*2007-02-26*/
+   JS_PAGE_FREE(item);// 2007/04/25
 }

 static JSHashEntry *
 AllocCallEntry(void *pool, const void *key)
 {
-    return (JSHashEntry*) calloc(1, sizeof(CallEntry));
+    //return (JSHashEntry*) SafeCalloc(1, sizeof(CallEntry));/*2007-02-26*/
+   return (JSHashEntry*) JS_PAGE_CALLOC(1, sizeof(CallEntry));// 2007/04/25
 }

 static void
 FreeCallEntry(void *pool, JSHashEntry *he, uintN flag)
 {
     JS_ASSERT(flag == HT_FREE_ENTRY);
-    free(he);
+    //SafeFree(he);/*2007-02-26*/
+   JS_PAGE_FREE(he);// 2007/04/25
 }

 static JSHashAllocOps callTableAllocOps = {
@@ -2337,13 +2346,24 @@
             ok = OBJ_LOOKUP_PROPERTY(cx, origobj, rval, &obj2, &prop);
             if (!ok)
                 goto out;
-            if (prop) {
+
+//Bug-list HDTV-BML No00239 START
+//            if (prop) {
+//                OBJ_DROP_PROPERTY(cx, obj2, prop);
+//
+//                /* Yes, don't enumerate again.  Go to the next property. */
+//                if (obj2 != obj)
+//                    goto enum_next_property;
+//            }
+            if (prop)
                 OBJ_DROP_PROPERTY(cx, obj2, prop);

-                /* Yes, don't enumerate again.  Go to the next property. */
-                if (obj2 != obj)
-                    goto enum_next_property;
+            /* idが削除済みの場合は、次のプロパティへスキップする */
+            /* Yes, don't enumerate again.  Go to the next property. */
+            if (!prop || (obj2 != obj)) {
+                goto enum_next_property;
             }
+//Bug-list HDTV-BML No00239 END

             /* Make sure rval is a string for uniformity and compatibility. */
             if (!JSVAL_IS_INT(rval)) {
@@ -2560,6 +2580,12 @@
     ((JSDOUBLE_IS_NaN(LVAL) || JSDOUBLE_IS_NaN(RVAL))                         ¥
      ? (IFNAN)                                                                ¥
     : (LVAL) OP (RVAL))
+/* bug 644 edit 2007/05/15*/
+#elif defined(_ROSA)
+#define COMPARE_DOUBLES(LVAL, OP, RVAL, IFNAN)                                ¥
+    ((JSDOUBLE_IS_NaN(LVAL) || JSDOUBLE_IS_NaN(RVAL))                         ¥
+    ? (IFNAN)                                                                ¥
+    : (LVAL) OP (RVAL))
 #else
 #define COMPARE_DOUBLES(LVAL, OP, RVAL, IFNAN) ((LVAL) OP (RVAL))
 #endif
@@ -2828,12 +2854,29 @@
                 if (d == 0 || JSDOUBLE_IS_NaN(d))
                     rval = DOUBLE_TO_JSVAL(rt->jsNaN);
                 else if ((JSDOUBLE_HI32(d) ^ JSDOUBLE_HI32(d2)) >> 31)
-                    rval = DOUBLE_TO_JSVAL(rt->jsNegativeInfinity);
+// Bug-list HDTV-BML No00233 MOD-START
+//                    rval = DOUBLE_TO_JSVAL(rt->jsNegativeInfinity);
+                    rval = DOUBLE_TO_JSVAL(rt->jsNaN);
+// Bug-list HDTV-BML No00233 MOD-END
                 else
-                    rval = DOUBLE_TO_JSVAL(rt->jsPositiveInfinity);
+// Bug-list HDTV-BML No00233 MOD-START
+//                    rval = DOUBLE_TO_JSVAL(rt->jsPositiveInfinity);
+                    rval = DOUBLE_TO_JSVAL(rt->jsNaN);
+// Bug-list HDTV-BML No00233 MOD-END
                 STORE_OPND(-1, rval);
             } else {
                 d /= d2;
+//2006-09-15 fixed /*2007-02-26*/
+// Bug-list HDTV-BML No00233 ADD-START
+//                d = (int)d;
+                if( !JSDOUBLE_IS_NaN( d ) )
+                {
+                    int32   ival;
+
+                    js_DoubleToECMAInt32( cx, d, &ival );
+                    d = (jsdouble)ival;
+// Bug-list HDTV-BML No00233 ADD-END
+                }
                 STORE_NUMBER(cx, -1, d);
```

```
             }
             break;
@@ -2954,18 +2997,31 @@
             rval = *vp;
             if (JSVAL_IS_PRIMITIVE(rval)) {
                 if (fun || !JSVERSION_IS_ECMA(cx->version)) {
-                    *vp = OBJECT_TO_JSVAL(obj);
-                    break;
-                }
-                /* native [[Construct]] returning primitive is error */
-                str = js_ValueToString(cx, rval);
-                if (str) {
-                    JS_ReportErrorNumber(cx, js_GetErrorMessage, NULL,
-                                         JSMSG_BAD_NEW_RESULT,
-                                         JS_GetStringBytes(str));
+                   /*bug#659
+                   (arib-std B-24 v5.)
+                   Constructor of BinaryTable object
+                   Return values
+                       BinaryTable object generated: Success
+                       null: Generation failed
+                   2007/05/12
+                   */
+                   if(0 != strcmp(fun->clasp->name,"BinaryTable"))
+                   {
+                       *vp = OBJECT_TO_JSVAL(obj);
+                       break;
+                   }
                 }
-                ok = JS_FALSE;
-                goto out;
+                /* native [[Construct]] returning primitive is error */
+//2006-09-15 fixed
+               /*
+               str = JS_ValueToString(cx , rval);
+               if (str)
+               {
+                   S_ReportErrorNumber(cx , JS_GetErrorMessage , NULL , JSMSG_BAD_NEW_RESULT , S_GetStringBytes(str));
+               }
+               ok = S_FALSE;
+               goto out;
+               */
             }
             obj = JSVAL_TO_OBJECT(rval);
             JS_RUNTIME_METER(rt, constructs);
@@ -3007,6 +3063,22 @@
           case JSOP_TYPEOF:
             rval = POP_OPND();
             type = JS_TypeOfValue(cx, rval);
+// Bug-list HDTV-BML No00237 ADD-START
+            if( (rval != NULL) && (type == JSTYPE_OBJECT) ) {
+                JSClass *objclass;
+                objclass = OBJ_GET_CLASS( cx, JSVAL_TO_OBJECT(rval) );
+                // hostobjectか判定
+                if( ( strcmp( objclass->name, "browser"  ) == 0 ) ||
+                    ( strcmp( objclass->name, "document" ) == 0 ) ||
+                    ( strcmp( objclass->name, "Ureg"     ) == 0 ) )
+                {
+                    atom = rt->atomState.hostobjectAtom;
+                    str = ATOM_TO_STRING(atom);
+                    PUSH_OPND(STRING_TO_JSVAL(str));
+                    break;
+                }
+            }
+// Bug-list HDTV-BML No00237 ADD-END
             atom = rt->atomState.typeAtoms[type];
             str  = ATOM_TO_STRING(atom);
             PUSH_OPND(STRING_TO_JSVAL(str));
@@ -3221,6 +3293,9 @@
           case JSOP_SETELEM:
             rval = FETCH_OPND(-1);
             ELEMENT_OP(-2, CACHED_SET(OBJ_SET_PROPERTY(cx, obj, id, &rval)));
+            //Bug-list HDTV-BML No00266 START
+            CheckObjGreg(cx, obj);
+            //Bug-list HDTV-BML No00266 END
             sp -= 2;
             STORE_OPND(-1, rval);
             break;
@@ -3425,6 +3500,7 @@
 #endif

           case JSOP_NAME:
+#if 0
             atom = GET_ATOM(cx, script, pc);
             id   = (jsid)atom;

@@ -3445,7 +3521,56 @@
                 }
                 goto atom_not_defined;
             }
-
+
+#else
+           {
+           /**
+           try to solve reference error.
+           1. if a name represents a variable, i.e. not a function
+           2. then set it to undefined (Underfined type with undefined value)
+
```

```
+             does it work? only through testing that we can tell.
+
+               -- 2007/05/12
+               */
+               JSBool first_try = JS_TRUE;
+second_chance:
+
+               atom = GET_ATOM(cx, script, pc);
+               id   = (jsid)atom;
+
+               SAVE_SP(fp);
+
+               ok = js_FindProperty(cx, id, &obj, &obj2, &prop);
+               if (!ok)
+                   goto out;
+               if (!prop) {
+                   /* Kludge to allow (typeof foo == "undefined") tests. */
+                   for (pc2 = pc + len; pc2 < endpc; pc2++) {
+                       op2 = (JSOp)*pc2;
+                       if (op2 == JSOP_TYPEOF) {
+                           PUSH_OPND(JSVAL_VOID);
+                           goto advance_pc;
+                       }
+                       if (op2 != JSOP_GROUP)
+                           break;
+                   }
+                   /*
+                   reference error issue,
+                   it seems that if a name represents a function, then it is followed by op2 (JSOP_PUSHOBJ).
+                   -- 2007/05/12
+                   */
+                   if (op2 != JSOP_PUSHOBJ && first_try) {
+                       attrs = JSPROP_ENUMERATE;
+                       ok = OBJ_DEFINE_PROPERTY(cx, obj, id, JSVAL_VOID, NULL, NULL, attrs, &prop);
+                       first_try = JS_FALSE;
+
+                       goto second_chance;
+                   }
+
+                   goto atom_not_defined;
+               }
+           }
+#endif
            /* Take the slow path if prop was not found in a native object. */
            if (!OBJ_IS_NATIVE(obj) || !OBJ_IS_NATIVE(obj2)) {
                OBJ_DROP_PROPERTY(cx, obj2, prop);
diff -u -d -r -N JS1.5ORG/jslock.c JS1.5MOD/jslock.c
--- JS1.5ORG/jslock.c   2004-04-04 07:21:02.000000000 +0900
+++ JS1.5MOD/jslock.c   2011-03-04 16:32:48.000000000 +0900
@@ -197,7 +197,8 @@

 #ifdef DEBUG_SCOPE_COUNT

-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include "jsdhash.h"

 static FILE *logfp;
@@ -709,7 +710,8 @@
 static JSFatLock *
 NewFatlock()
 {
-    JSFatLock *fl = (JSFatLock *)malloc(sizeof(JSFatLock)); /* for now */
+    //JSFatLock *fl = (JSFatLock *)SafeMalloc(sizeof(JSFatLock)); /* for now *//*2007-02-26*/
+   JSFatLock *fl = (JSFatLock *)JS_PAGE_MALLOC(sizeof(JSFatLock)); // 2007/04/25
     if (!fl) return NULL;
     fl->susp = 0;
     fl->next = NULL;
@@ -724,7 +726,8 @@
 {
     PR_DestroyLock(fl->slock);
     PR_DestroyCondVar(fl->svar);
-    free(fl);
+    //SafeFree(fl);/*2007-02-26*/
+   JS_PAGE_FREE(fl);// 2007/04/25
 }

 static JSFatLock *
@@ -818,7 +821,8 @@
     global_locks_log2 = JS_CeilingLog2(globc);
     global_locks_mask = JS_BITMASK(global_locks_log2);
     global_lock_count = JS_BIT(global_locks_log2);
-    global_locks = (PRLock **) malloc(global_lock_count * sizeof(PRLock*));
+    //global_locks = (PRLock **) SafeMalloc(global_lock_count * sizeof(PRLock*));/*2007-02-26*/
+   global_locks = (PRLock **) JS_PAGE_MALLOC(global_lock_count * sizeof(PRLock*));// 2007/04/25
     if (!global_locks)
         return JS_FALSE;
     for (i = 0; i < global_lock_count; i++) {
@@ -829,7 +833,8 @@
             return JS_FALSE;
         }
     }
-    fl_list_table = (JSFatLockTable *) malloc(i * sizeof(JSFatLockTable));
+    //fl_list_table = (JSFatLockTable *) SafeMalloc(i * sizeof(JSFatLockTable));/*2007-02-26*/
+   fl_list_table = (JSFatLockTable *) JS_PAGE_MALLOC(i * sizeof(JSFatLockTable));// 2007/04/25
     if (!fl_list_table) {
         js_CleanupLocks();
```

```
         return JS_FALSE;
@@ -851,7 +856,8 @@
     if (global_locks) {
         for (i = 0; i < global_lock_count; i++)
             PR_DestroyLock(global_locks[i]);
-        free(global_locks);
+        //SafeFree(global_locks);/*2007-02-26*/
+       JS_PAGE_FREE(global_locks);// 2007/04/25
         global_locks = NULL;
         global_lock_count = 1;
         global_locks_log2 = 0;
@@ -864,7 +870,8 @@
             DeleteListOfFatlocks(fl_list_table[i].taken);
             fl_list_table[i].taken = NULL;
         }
-        free(fl_list_table);
+        //SafeFree(fl_list_table);/*2007-02-26*/
+       JS_PAGE_FREE(fl_list_table);// 2007/04/25
         fl_list_table = NULL;
         fl_list_table_len = 0;
     }
diff -u -d -r -N JS1.5ORG/jsnum.c JS1.5MOD/jsnum.c
--- JS1.5ORG/jsnum.c    2004-07-17 11:44:36.000000000 +0900
+++ JS1.5MOD/jsnum.c    2011-03-04 16:33:12.000000000 +0900
@@ -45,7 +45,9 @@
 #if defined(XP_WIN) || defined(XP_OS2)
 #include <float.h>
 #endif
+#if !defined(_ROSA)/*2007-02-26*/
 #include <locale.h>
+#endif
 #include <limits.h>
 #include <math.h>
 #include <stdlib.h>
@@ -69,10 +71,20 @@
 num_isNaN(JSContext *cx, JSObject *obj, uintN argc, jsval *argv, jsval *rval)
 {
     jsdouble x;
-
+
     if (!js_ValueToNumber(cx, argv[0], &x))
+   {
         return JS_FALSE;
-    *rval = BOOLEAN_TO_JSVAL(JSDOUBLE_IS_NaN(x));
+   }
+#if 0  /* fix by yanjq 20070402 for num_isNaN() */
+   if (JSDOUBLE_LO32(x) == 0xFFFFFFFF)
+       *rval = BOOLEAN_TO_JSVAL(1);
+   else
+       *rval = BOOLEAN_TO_JSVAL(0);
+#else
+   *rval = BOOLEAN_TO_JSVAL(JSDOUBLE_IS_NaN(x));
+#endif
+
     return JS_TRUE;
 }

@@ -282,7 +294,8 @@
             return JS_FALSE;
         }
         str = JS_NewStringCopyZ(cx, dStr);
-        free(dStr);
+        //SafeFree(dStr);/*2007-02-26*/
+       JS_PAGE_FREE(dStr);//2007/04/25
     }
     if (!str)
         return JS_FALSE;
@@ -496,8 +509,18 @@
     {0,                         js_NaN_str,          0,{0,0,0}},
     {0,                         "POSITIVE_INFINITY", 0,{0,0,0}},
     {0,                         "NEGATIVE_INFINITY", 0,{0,0,0}},
+/* 2007/08/19
+refer to "ARIB STD-B24 Version 5.0-E1 Vol2 (2/2) pp.531"
+- MAX_VALUE attribute must be 2147483647. (Maximum value of signed 32-bit integer)
+- MIN_VALUE attribute must be 1. (Minimum value of signed 32-bit integer)
+*/
+#if 1
+    {2147483647,    "MAX_VALUE",         0,{0,0,0}},
+    {1,             "MIN_VALUE",         0,{0,0,0}},
+#else
     {1.7976931348623157E+308,   "MAX_VALUE",         0,{0,0,0}},
     {0,                         "MIN_VALUE",         0,{0,0,0}},
+#endif
     {0,0,0,{0,0,0}}
 };

@@ -559,10 +582,15 @@
         return JS_FALSE;
     }

+/* 2007/08/19
+refer to "ARIB STD-B24 Version 5.0-E1 Vol2 (2/2) pp.531"
+- MIN_VALUE attribute must be 1. (Minimum value of signed 32-bit integer)
     u.s.hi = 0;
     u.s.lo = 1;
     number_constants[NC_MIN_VALUE].dval = u.d;
+*/
```

```
+#if !defined(_ROSA) /*add 2007-02-26*/
     locale = localeconv();
     rt->thousandsSeparator =
         JS_strdup(cx, locale->thousands_sep ? locale->thousands_sep : "'");
@@ -572,6 +600,9 @@
         JS_strdup(cx, locale->grouping ? locale->grouping : "¥3¥0");

     return rt->thousandsSeparator && rt->decimalSeparator && rt->numGrouping;
+#endif
+    return JS_TRUE;
+
 }

 void
diff -u -d -r -N JS1.5ORG/jsnum.h JS1.5MOD/jsnum.h
--- JS1.5ORG/jsnum.h    2004-04-01 01:38:54.000000000 +0900
+++ JS1.5MOD/jsnum.h    2011-03-04 15:25:54.000000000 +0900
@@ -66,7 +66,8 @@

 typedef union jsdpun {
     struct {
-#if defined(IS_LITTLE_ENDIAN) && !defined(CPU_IS_ARM)
+//#if defined(IS_LITTLE_ENDIAN) && !defined(CPU_IS_ARM)
+#if defined(IS_LITTLE_ENDIAN)
     uint32 lo, hi;
 #else
     uint32 hi, lo;
@@ -95,7 +96,8 @@
  * so this code should work.
  */

-#if defined(IS_LITTLE_ENDIAN) && !defined(CPU_IS_ARM)
+//#if defined(IS_LITTLE_ENDIAN) && !defined(CPU_IS_ARM)
+#if defined(IS_LITTLE_ENDIAN)
 #define JSDOUBLE_HI32(x)        (((uint32 *)&(x))[1])
 #define JSDOUBLE_LO32(x)        (((uint32 *)&(x))[0])
 #else
@@ -112,9 +114,11 @@
 #define JSDOUBLE_HI32_EXPMASK   0x7ff00000
 #define JSDOUBLE_HI32_MANTMASK  0x000fffff

-#define JSDOUBLE_IS_NaN(x)                                              ¥
-    ((JSDOUBLE_HI32(x) & JSDOUBLE_HI32_EXPMASK) == JSDOUBLE_HI32_EXPMASK &&   ¥
-     (JSDOUBLE_LO32(x) || (JSDOUBLE_HI32(x) & JSDOUBLE_HI32_MANTMASK)))
+
+#define JSDOUBLE_IS_NaN(x)                                             ¥
+    ((JSDOUBLE_HI32(x)& JSDOUBLE_HI32_EXPMASK) == JSDOUBLE_HI32_EXPMASK &&   ¥
+     (JSDOUBLE_LO32(x)|| (JSDOUBLE_HI32(x)& JSDOUBLE_HI32_MANTMASK)))
+

 #define JSDOUBLE_IS_INFINITE(x)                                         ¥
     ((JSDOUBLE_HI32(x) & ~JSDOUBLE_HI32_SIGNBIT) == JSDOUBLE_HI32_EXPMASK &&  ¥
@@ -122,9 +126,9 @@

 #define JSDOUBLE_IS_FINITE(x)                                           ¥
     ((JSDOUBLE_HI32(x) & JSDOUBLE_HI32_EXPMASK) != JSDOUBLE_HI32_EXPMASK)
-
-#define JSDOUBLE_IS_NEGZERO(d)  (JSDOUBLE_HI32(d) == JSDOUBLE_HI32_SIGNBIT && ¥
-                JSDOUBLE_LO32(d) == 0)
+//changed 2009/12/11 for QAC [MISRA Rule 47]
+#define JSDOUBLE_IS_NEGZERO(d)  ((JSDOUBLE_HI32(d) == JSDOUBLE_HI32_SIGNBIT) && ¥
+                (JSDOUBLE_LO32(d) == 0))

 /*
  * JSDOUBLE_IS_INT first checks that d is neither NaN nor infinite, to avoid
diff -u -d -r -N JS1.5ORG/jsobj.c JS1.5MOD/jsobj.c
--- JS1.5ORG/jsobj.c    2004-09-02 05:51:38.000000000 +0900
+++ JS1.5MOD/jsobj.c    2011-03-04 15:15:40.000000000 +0900
@@ -660,7 +660,8 @@

     if (!chars) {
         /* If outermost, allocate 4 + 1 for "({})" and the terminator. */
-        chars = (jschar *) malloc(((outermost ? 4 : 2) + 1) * sizeof(jschar));
+        //chars = (jschar *) SafeMalloc(((outermost ? 4 : 2) + 1) * sizeof(jschar));/*2007-02-26*/
+       chars = (jschar *) JS_PAGE_MALLOC(((outermost ? 4 : 2) + 1) * sizeof(jschar));// 2007/04/25
         nchars = 0;
         if (!chars)
             goto error;
@@ -671,9 +672,11 @@
         MAKE_SHARP(he);
         nchars = js_strlen(chars);
         chars = (jschar *)
-            realloc((ochars = chars), (nchars + 2 + 1) * sizeof(jschar));
+            //SafeRealloc((ochars = chars), (nchars + 2 + 1) * sizeof(jschar));/*2007-02-26*/
+           JS_PAGE_REALLOC((ochars = chars), (nchars + 2 + 1) * sizeof(jschar));// 2007/04/04
         if (!chars) {
-            free(ochars);
+            //SafeFree(ochars);/*2007-02-26*/
+           JS_PAGE_FREE(ochars);// 2007/04/25
             goto error;
         }
         if (outermost) {
@@ -701,10 +704,12 @@

         /* 2 for the braces, 1 for the terminator */
         chars = (jschar *)
-            realloc((ochars = chars),
+            //SafeRealloc((ochars = chars),/*2007-02-26*/
```

```
+           JS_PAGE_REALLOC((ochars = chars),// 2007/04/25
                    (nchars + classnchars + 2 + 1) * sizeof(jschar));
         if (!chars) {
-            free(ochars);
+            //SafeFree(ochars);/*2007-02-26*/
+           JS_PAGE_FREE(ochars);// 2007/04/25
             goto error;
         }

@@ -862,7 +867,8 @@

             /* Allocate 1 + 1 at end for closing brace and terminating 0. */
             chars = (jschar *)
-                realloc((ochars = chars),
+                //SafeRealloc((ochars = chars),/*2007-02-26*/
+               JS_PAGE_REALLOC((ochars = chars),// 2007/04/24
                         (nchars + (comma ? 2 : 0) +
                          idstrlength + 1 +
                          (gsop[j] ? 1 + JSSTRING_LENGTH(gsop[j]) : 0) +
@@ -871,7 +877,8 @@
             if (!chars) {
                 /* Save code space on error: let JS_free ignore null vsharp. */
                 JS_free(cx, vsharp);
-                free(ochars);
+                //SafeFree(ochars);/*2007-02-26*/
+               JS_PAGE_FREE(ochars);// 2007/04/25
                 goto error;
             }

@@ -929,7 +936,8 @@

     if (!ok) {
         if (chars)
-            free(chars);
+            //SafeFree(chars);/*2007-02-26*/
+           JS_PAGE_FREE(chars);// 2007/04/25
         return ok;
     }

@@ -940,7 +948,8 @@
   make_string:
     str = js_NewString(cx, chars, nchars, 0);
     if (!str) {
-        free(chars);
+        //SafeFree(chars);/*2007-02-26*/
+       JS_PAGE_FREE(chars);// 2007/04/25
         return JS_FALSE;
     }
     *rval = STRING_TO_JSVAL(str);
@@ -3087,7 +3096,8 @@
                     return JS_FALSE;
                 str = JS_NewString(cx, bytes, strlen(bytes));
                 if (!str) {
-                    free(bytes);
+                    //SafeFree(bytes);/*2007-02-26*/
+                   JS_PAGE_FREE(bytes);// 2007/04/25
                     return JS_FALSE;
                 }
                 v = STRING_TO_JSVAL(str);
@@ -3750,7 +3760,8 @@

 #ifdef DEBUG_brendan

-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <math.h>

 uint32 js_entry_count_max;
diff -u -d -r -N JS1.5ORG/jsopcode.c JS1.5MOD/jsopcode.c
--- JS1.5ORG/jsopcode.c 2004-08-30 03:00:24.000000000 +0900
+++ JS1.5MOD/jsopcode.c 2011-03-04 16:34:10.000000000 +0900
@@ -45,7 +45,8 @@
 #include <memory.h>
 #endif
 #include <stdarg.h>
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <stdlib.h>
 #include <string.h>
 #include "jstypes.h"
@@ -79,6 +80,8 @@
 const char js_typeof_str[]      = "typeof";
 const char js_void_str[]        = "void";
 const char js_null_str[]        = "null";
+// Bug #610 we may have Null too in some cases, 2007/05/04
+const char js_null_str_arib[]       = "Null";
 const char js_this_str[]        = "this";
 const char js_false_str[]       = "false";
 const char js_true_str[]        = "true";
@@ -372,7 +375,8 @@
         return -1;
     }
     offset = SprintPut(sp, bp, strlen(bp));
-    free(bp);
+    //SafeFree(bp);/*2007-02-26*/
+   JS_PAGE_FREE(bp);// 2007/04/25
```

```
     return offset;
 }

@@ -567,7 +571,8 @@
     cc = strlen(bp);
     if (SprintPut(&jp->sprinter, bp, (size_t)cc) < 0)
         cc = -1;
-    free(bp);
+    //SafeFree(bp);/*2007-02-26*/
+   JS_PAGE_FREE(bp);// 2007/04/25

     va_end(ap);
     return cc;
@@ -2443,7 +2448,7 @@
      * an expression by parenthesizing.
      */
     if (jp->pretty) {
-        js_puts(jp, "¥n");
+//Bug-list HDTV-BML No00231 DEL//        js_puts(jp, "¥n");
         js_printf(jp, "¥t");
     } else {
         if (!jp->grouped && (fun->flags & JSFUN_LAMBDA))
@@ -2460,6 +2465,8 @@
     js_puts(jp, "(");

     if (fun->interpreted && fun->object) {
+//Bug-list HDTV-BML No00231 DEL-START
+#if 0  // ARIB STD B-24記載のフォーマットにあわせる対応
         /*
          * Print the parameters.
          *
@@ -2495,13 +2502,20 @@
                 return JS_FALSE;
         }
         JS_ARENA_RELEASE(&cx->tempPool, mark);
+#endif // ARIB STD B-24記載のフォーマットにあわせる対応
+//Bug-list HDTV-BML No00231 DEL-END
 #ifdef __GNUC__
     } else {
         scope = NULL;
 #endif
     }

-    js_printf(jp, ") {¥n");
+//Bug-list HDTV-BML No00231 START
+//    js_printf(jp, ") {¥n");
+    js_printf(jp, ") {}");
+//Bug-list HDTV-BML No00231 END
+//Bug-list HDTV-BML No00231 DEL-START
+#if 0  // ARIB STD B-24記載のフォーマットにあわせる対応
     indent = jp->indent;
     jp->indent += 4;
     if (fun->interpreted && fun->object) {
@@ -2518,9 +2532,11 @@
     }
     jp->indent -= 4;
     js_printf(jp, "¥t}");
+#endif // ARIB STD B-24記載のフォーマットにあわせる対応
+//Bug-list HDTV-BML No00231 DEL-END

     if (jp->pretty) {
-        js_puts(jp, "¥n");
+//Bug-list HDTV-BML No00231 DEL//        js_puts(jp, "¥n");
     } else {
         if (!jp->grouped && (fun->flags & JSFUN_LAMBDA))
             js_puts(jp, ")");
diff -u -d -r -N JS1.5ORG/jsopcode.h JS1.5MOD/jsopcode.h
--- JS1.5ORG/jsopcode.h 2004-09-02 05:51:38.000000000 +0900
+++ JS1.5MOD/jsopcode.h 2011-03-04 15:40:14.000000000 +0900
@@ -190,6 +190,8 @@
 extern const char       js_void_str[];
 extern const char       js_null_str[];
 extern const char       js_this_str[];
+// Bug #610 we may have Null too in some cases, 2007/05/04
+extern const char       js_null_str_arib[];
 extern const char       js_false_str[];
 extern const char       js_true_str[];
 extern const JSCodeSpec js_CodeSpec[];
@@ -228,7 +230,8 @@
 /*
  * Disassemblers, for debugging only.
  */
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"

 extern JS_FRIEND_API(void)
 js_Disassemble(JSContext *cx, JSScript *script, JSBool lines, FILE *fp);
diff -u -d -r -N JS1.5ORG/jsparse.c JS1.5MOD/jsparse.c
--- JS1.5ORG/jsparse.c  2004-09-14 10:38:18.000000000 +0900
+++ JS1.5MOD/jsparse.c  2011-03-04 15:29:32.000000000 +0900
@@ -3105,7 +3105,25 @@
         pn = NewParseNode(cx, &CURRENT_TOKEN(ts), PN_NULLARY, tc);
         if (!pn)
             return NULL;
-        pn->pn_dval = CURRENT_TOKEN(ts).t_dval;
+//2006-09-15 fixed /*2007-02-26*/
+//      pn->pn_dval = CURRENT_TOKEN(ts).t_dval;
```

```
+// Bug-list HDTV-BML No00181 START
+//     pn->pn_dval = (int)(CURRENT_TOKEN(ts).t_dval);
+        /* 負の値の場合は符号ありで変換し、負の値のまま格納する */
+        /*  16進数で記載された値の場合、単項演算子なしで負の値とする対応 */
+        if( (CURRENT_TOKEN(ts).t_dval) < 0.0 ) {
+            int32 ival;
+            js_DoubleToECMAInt32( cx, (CURRENT_TOKEN(ts).t_dval), &ival );
+            pn->pn_dval = (jsdouble)ival;
+        }
+        /* 正の値の場合は符号なしで変換し、正の値のまま格納する */
+        /*  単項演算子の'-'が付いている場合、他のタイミングで負の値へ変換される */
+        else {
+            uint32 uval;
+            js_DoubleToECMAUint32( cx, (CURRENT_TOKEN(ts).t_dval), &uval );
+            pn->pn_dval = (jsdouble)uval;
+        }
+// Bug-list HDTV-BML No00181 END
 #if JS_HAS_SHARP_VARS
         notsharp = JS_TRUE;
 #endif
@@ -3299,6 +3317,8 @@
                 d = *cx->runtime->jsPositiveInfinity;
         } else {
             d /= d2;
+//2006-09-15 fixed /*2007-02-26*/
+           d = (int)d;
         }
         break;

diff -u -d -r -N JS1.5ORG/jsprf.c JS1.5MOD/jsprf.c
--- JS1.5ORG/jsprf.c    2004-09-24 12:31:14.000000000 +0900
+++ JS1.5MOD/jsprf.c    2011-03-04 15:17:44.000000000 +0900
@@ -43,7 +43,8 @@
 */
 #include "jsstddef.h"
 #include <stdarg.h>
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <string.h>
 #include <stdlib.h>
 #include "jsprf.h"
@@ -452,7 +453,8 @@


     if( number > NAS_DEFAULT_NUM ){
-        nas = (struct NumArgState*)malloc( number * sizeof( struct NumArgState ) );
+        //nas = (struct NumArgState*)SafeMalloc( number * sizeof( struct NumArgState ) );/*2007-02-26*/
+       nas = (struct NumArgState*)JS_PAGE_MALLOC( number * sizeof( struct NumArgState ) );// 2007/04/25
         if( !nas ){
             *rv = -1;
             return NULL;
@@ -606,7 +608,8 @@

     if( *rv < 0 ){
         if( nas != nasArray )
-            free( nas );
+            //SafeFree( nas );/*2007-02-26*/
+           JS_PAGE_FREE(nas);// 2007/04/25
         return NULL;
     }

@@ -641,7 +644,8 @@

         default:
             if( nas != nasArray )
-                free( nas );
+                //SafeFree( nas );/*2007-02-26*/
+               JS_PAGE_FREE(nas);// 2007/04/25
             *rv = -1;
             return NULL;
         }
@@ -727,7 +731,8 @@

             if( nas[i-1].type == TYPE_UNKNOWN ){
                 if( nas && ( nas != nasArray ) )
-                    free( nas );
+                    //SafeFree( nas );/*2007-02-26*/
+                   JS_PAGE_FREE(nas);// 2007/04/25
                 return -1;
             }

@@ -987,7 +992,8 @@
     rv = (*ss->stuff)(ss, "¥0", 1);

     if( nas && ( nas != nasArray ) ){
-        free( nas );
+        //SafeFree( nas );/*2007-02-26*/
+       JS_PAGE_FREE(nas);// 2007/04/25
     }

     return rv;
@@ -1048,9 +1054,11 @@
         /* Grow the buffer */
         newlen = ss->maxlen + ((len > 32) ? len : 32);
         if (ss->base) {
-            newbase = (char*) realloc(ss->base, newlen);
+            //newbase = (char*) SafeRealloc(ss->base, newlen);/*2007-02-26*/
```

```
+           newbase = (char*) JS_PAGE_REALLOC(ss->base, newlen);// 2007/04/25
         } else {
-            newbase = (char*) malloc(newlen);
+            //newbase = (char*) SafeMalloc(newlen);/*2007-02-26*/
+           newbase = (char*) JS_PAGE_MALLOC(newlen);// 2007/04/25
         }
         if (!newbase) {
             /* Ran out of memory */
@@ -1089,7 +1097,8 @@
 */
 JS_PUBLIC_API(void) JS_smprintf_free(char *mem)
 {
-        free(mem);
+        //SafeFree(mem);/*2007-02-26*/
+       JS_PAGE_FREE(mem);// 2007/04/25
 }

 JS_PUBLIC_API(char *) JS_vsmprintf(const char *fmt, va_list ap)
@@ -1104,7 +1113,8 @@
     rv = dosprintf(&ss, fmt, ap);
     if (rv < 0) {
         if (ss.base) {
-            free(ss.base);
+            //SafeFree(ss.base);/*2007-02-26*/
+           JS_PAGE_FREE(ss.base);// 2007/04/25
         }
         return 0;
     }
@@ -1203,7 +1213,8 @@
     rv = dosprintf(&ss, fmt, ap);
     if (rv < 0) {
         if (ss.base) {
-            free(ss.base);
+            //SafeFree(ss.base);/*2007-02-26*/
+           JS_PAGE_FREE(ss.base);// 2007/04/25
         }
         return 0;
     }
diff -u -d -r -N JS1.5ORG/jsprf.h JS1.5MOD/jsprf.h
--- JS1.5ORG/jsprf.h    2003-11-15 09:10:58.000000000 +0900
+++ JS1.5MOD/jsprf.h    2011-03-04 15:26:16.000000000 +0900
@@ -56,7 +56,8 @@
 ** %g - float
 */
 #include "jstypes.h"
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <stdarg.h>

 JS_BEGIN_EXTERN_C
diff -u -d -r -N JS1.5ORG/jspubtd.h JS1.5MOD/jspubtd.h
--- JS1.5ORG/jspubtd.h  2004-06-16 01:38:42.000000000 +0900
+++ JS1.5MOD/jspubtd.h  2011-03-04 15:26:28.000000000 +0900
@@ -513,8 +513,8 @@
 (* JS_DLL_CALLBACK JSBranchCallback)(JSContext *cx, JSScript *script);

 typedef void
-(* JS_DLL_CALLBACK JSErrorReporter)(JSContext *cx, const char *message,
-                                    JSErrorReport *report);
+(* JS_DLL_CALLBACK JSErrorReporter)(JSContext *context, const char *message,
+                                    JSErrorReport *report);//change 2009/12/22 for QAC [MISRA Rule 74]

 typedef struct JSErrorFormatString {
     const char *format;
diff -u -d -r -N JS1.5ORG/jsscan.c JS1.5MOD/jsscan.c
--- JS1.5ORG/jsscan.c   2004-08-25 11:27:24.000000000 +0900
+++ JS1.5MOD/jsscan.c   2011-03-04 16:35:14.000000000 +0900
@@ -41,7 +41,8 @@
  * JS lexical scanner.
  */
 #include "jsstddef.h"
-#include <stdio.h>      /* first to avoid trouble on some systems */
+/*#include <stdio.h> 2007-02-26*/     /* first to avoid trouble on some systems */
+#include "ebml_fio.h"
 #include <errno.h>
 #include <limits.h>
 #include <math.h>
@@ -91,6 +92,8 @@
     {js_in_str,         TOK_IN,                 JSOP_IN,    JSVERSION_DEFAULT},
     {js_new_str,        TOK_NEW,                JSOP_NEW,   JSVERSION_DEFAULT},
     {js_null_str,       TOK_PRIMARY,            JSOP_NULL,  JSVERSION_DEFAULT},
+   // Bug #610 we may have Null too in some cases, 2007/05/04
+    {js_null_str_arib,   TOK_PRIMARY,            JSOP_NULL,  JSVERSION_DEFAULT},
     {"return",          TOK_RETURN,             JSOP_NOP,   JSVERSION_DEFAULT},
     {"switch",          TOK_SWITCH,             JSOP_NOP,   JSVERSION_DEFAULT},
     {js_this_str,       TOK_PRIMARY,            JSOP_THIS,  JSVERSION_DEFAULT},
@@ -249,7 +252,7 @@
     if (!filename || strcmp(filename, "-") == 0) {
         file = defaultfp;
     } else {
-        file = fopen(filename, "r");
+        file = (FILE *)fopen(filename, "r");
         if (!file) {
             JS_ReportErrorNumber(cx, js_GetErrorMessage, NULL, JSMSG_CANT_OPEN,
                                  filename, "No such file or directory");
@@ -269,7 +272,7 @@
         JS_free(cx, (void *) ts->filename);
```

```
     if (ts->principals)
         JSPRINCIPALS_DROP(cx, ts->principals);
-    return !ts->file || fclose(ts->file) == 0;
+   return !ts->file || fclose((VP)ts->file) == 0;
 }

 static int
@@ -283,14 +286,14 @@
         return -1;

     crflag = JS_FALSE;
-    for (i = 0; i < n && (c = getc(file)) != EOF; i++) {
+   for (i = 0; i < n && (c = getc((VP)file)) != EOF; i++) {
         buf[i] = c;
         if (c == '¥n') {        /* any ¥n ends a line */
             i++;                /* keep the ¥n; we know there is room for ¥0 */
             break;
         }
         if (crflag) {           /* ¥r not followed by ¥n ends line at the ¥r */
-            ungetc(c, file);
+           ungetc(c, (VP)file);
             break;              /* and overwrite c in buf with ¥0 */
         }
         crflag = (c == '¥r');
@@ -936,6 +939,18 @@
                 goto error;
             }
         }
+// Bug-list HDTV-BML No00181 START
+        // 入力される文字が16進数表現の場合、数値へ変換したjsdouble型の結果を32bit変換した後に、
+        // 最上位ビットが1足っていた場合は（32bit表現で負の場合）、負の値としてjsdouble型の変数へ格納する処理を追加。
+        if( radix == 16 ) {
+            int32 ival;
+            js_DoubleToECMAInt32( cx, dval, &ival );
+            if( (ival & 0x80000000) != 0 ) {
+                dval = (jsdouble)ival;
+            }
+        }
+// Bug-list HDTV-BML No00181 END
+
         tp->t_dval = dval;
         tt = TOK_NUMBER;
         goto out;
diff -u -d -r -N JS1.5ORG/jsscan.h JS1.5MOD/jsscan.h
--- JS1.5ORG/jsscan.h   2004-08-25 11:27:24.000000000 +0900
+++ JS1.5MOD/jsscan.h   2011-03-04 15:26:38.000000000 +0900
@@ -43,7 +43,8 @@
  * JS lexical scanner interface.
  */
 #include <stddef.h>
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include "jsopcode.h"
 #include "jsprvtd.h"
 #include "jspubtd.h"
diff -u -d -r -N JS1.5ORG/jsscope.c JS1.5MOD/jsscope.c
--- JS1.5ORG/jsscope.c  2004-04-13 10:25:16.000000000 +0900
+++ JS1.5MOD/jsscope.c  2011-03-04 15:19:16.000000000 +0900
@@ -119,7 +119,8 @@
     }

     scope->table = (JSScopeProperty **)
-        calloc(JS_BIT(sizeLog2), sizeof(JSScopeProperty *));
+        //SafeCalloc(JS_BIT(sizeLog2), sizeof(JSScopeProperty *));/*2007-02-26*/
+       JS_PAGE_CALLOC(JS_BIT(sizeLog2), sizeof(JSScopeProperty *));// 2007/04/25
     if (!scope->table)
         return JS_FALSE;

@@ -335,7 +336,8 @@
     oldsize = JS_BIT(oldlog2);
     newsize = JS_BIT(newlog2);
     nbytes = SCOPE_TABLE_NBYTES(newsize);
-    table = (JSScopeProperty **) calloc(nbytes, 1);
+    //table = (JSScopeProperty **) SafeCalloc(nbytes, 1);/*2007-02-26*/
+   table = (JSScopeProperty **) JS_PAGE_CALLOC(nbytes, 1);// 2007/04/25
     if (!table) {
         JS_ReportOutOfMemory(cx);
         return JS_FALSE;
@@ -510,7 +512,8 @@
 {
     PropTreeKidsChunk *chunk;

-    chunk = calloc(1, sizeof *chunk);
+    //chunk = SafeCalloc(1, sizeof *chunk);/*2007-02-26*/
+   chunk = JS_PAGE_CALLOC(1, sizeof *chunk);// 2007/04/25
     if (!chunk)
         return NULL;
     JS_ASSERT(((jsuword)chunk & CHUNKY_KIDS_TAG) == 0);
@@ -522,7 +525,8 @@
 {
     JS_RUNTIME_UNMETER(rt, propTreeKidsChunks);
-    free(chunk);
+    //SafeFree(chunk);/*2007-02-26*/
+   JS_PAGE_FREE(chunk);// 2007/04/25
 }
```

```
 /* NB: Called with the runtime lock held. */
@@ -1367,14 +1371,16 @@
 #endif

     if (scope->table)
-        free(scope->table);
+        //SafeFree(scope->table);/*2007-02-26*/
+       JS_PAGE_FREE(scope->table);// 2007/04/25
     SCOPE_CLR_MIDDLE_DELETE(scope);
     InitMinimalScope(scope);
 }

 #ifdef DUMP_SCOPE_STATS

-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <math.h>

 uint32 js_nkids_max;
diff -u -d -r -N JS1.5ORG/jsscript.c JS1.5MOD/jsscript.c
--- JS1.5ORG/jsscript.c 2004-08-20 02:57:36.000000000 +0900
+++ JS1.5MOD/jsscript.c 2011-03-04 15:19:48.000000000 +0900
@@ -908,7 +908,8 @@
 {
     size_t nbytes = offsetof(ScriptFilenameEntry, filename) + strlen(key) + 1;

-    return (JSHashEntry *) malloc(JS_MAX(nbytes, sizeof(JSHashEntry)));
+    //return (JSHashEntry *) SafeMalloc(JS_MAX(nbytes, sizeof(JSHashEntry)));/*2007-02-26*/
+   return (JSHashEntry *) JS_PAGE_MALLOC(JS_MAX(nbytes, sizeof(JSHashEntry)));// 2007/04/25
 }

 JS_STATIC_DLL_CALLBACK(void)
@@ -916,7 +917,8 @@
 {
     if (flag != HT_FREE_ENTRY)
         return;
-    free(he);
+    //SafeFree(he);/*2007-02-26*/
+   JS_PAGE_FREE(he);// 2007/04/25
 }

 static JSHashAllocOps table_alloc_ops = {
diff -u -d -r -N JS1.5ORG/jsstr.c JS1.5MOD/jsstr.c
--- JS1.5ORG/jsstr.c    2003-12-22 15:13:06.000000000 +0900
+++ JS1.5MOD/jsstr.c    2011-03-04 15:45:00.000000000 +0900
@@ -223,7 +223,8 @@
     if (JSSTRING_IS_DEPENDENT(str)) {
         n = JSSTRDEP_LENGTH(str);
         size = (n + 1) * sizeof(jschar);
-        s = (jschar *) (cx ? JS_malloc(cx, size) : malloc(size));
+        //s = (jschar *) (cx ? JS_malloc(cx, size) : SafeMalloc(size));/*2007-02-26*/
+       s = (jschar *) (cx ? JS_malloc(cx, size) : JS_PAGE_MALLOC(size));// 2007/04/25
         if (!s)
             return NULL;

@@ -311,6 +312,90 @@

 #define IS_OK(C, mask) (urlCharType[((uint8) (C))] & (mask))

+/* to support EUC string
+  given an jschar index in JSString, return EUC char index
+  --- 2007/04/10
+*/
+static jsint js_fix_FromCharCharIndexToEucIndex(JSString *str, jsint char_index)
+{
+   const unsigned char* p;
+   jsint index = -1;
+
+   if (char_index == 0)
+   {
+       return 0;
+   }
+   else if (char_index < 0)
+   {
+       return index;
+   }
+
+   p = (unsigned char*)js_GetStringBytes(str);
+   if (p)
+   {
+       jsint i = 0;
+       jsint j = 0;
+       for (i = 0, index = 0, j = 0; j< char_index && *p != 0; i++)
+       {
+           if (*p++ < 0x7F)
+           {
+               // single byte
+               j++;
+           }
+           else
+           {
+               j += 2;
+               // double byte
+               p++;
+           }
+           index++;
+       }
```

```
+
+   }
+    return index;
+
+}
+
+/* to support EUC string
+  given an EUC char indexa, return jschar index in JSString
+  --- 2007/03/28
+*/
+static size_t js_fix_FromEucCharIndexToCharIndex(JSString *str, size_t euc_index)
+{
+   const unsigned char* p;
+   size_t index = -1;
+
+   if (euc_index == 0)
+   {
+       return 0;
+   }
+   else if (euc_index < 0)
+   {
+       return index;
+   }
+
+   p = (unsigned char*)js_GetStringBytes(str);
+   if (p)
+   {
+       size_t i = 0;
+       for (i = 0, index = 0; i < euc_index && *p != 0; i++)
+       {
+           if (*p++ < 0x7F) // single byte
+           {
+               index++;
+           }
+           else // double byte
+           {
+               index += 2;
+
+               p++;
+           }
+       }
+   }
+
+   return index;
+}
+
 /* See ECMA-262 15.1.2.4. */
 JSBool
 js_str_escape(JSContext *cx, JSObject *obj, uintN argc, jsval *argv, jsval *rval)
@@ -516,12 +601,60 @@
     if (!str)
         return JS_FALSE;
     slot = JSVAL_TO_INT(id);
-    if (slot == STRING_LENGTH)
-        *vp = INT_TO_JSVAL((jsint) JSSTRING_LENGTH(str));
+    if (slot == STRING_LENGTH)
+   {
+       /* to get EUC string length.
+
+         problem: "漢字123".length
+            expected 5, but returns 7 (漢--2, 字--2, 1--1, 2--1, 3--1)
+            BugZilla #331
+
+         EUC encoding
+        one byte or code set 0
+            byte range 21 ~ 7E
+        two bytes or code set 1
+            1st byte range A1~FE
+            2nd byte range A1~FE
+
+           --- 2007/03/28
+        */
+       const unsigned char * p = (unsigned char *)js_GetStringBytes(str);
+
+       size_t size = 0;
+       if (p)
+       {
+           while (*p)
+           {
+               size++;
+               if (*p++ > 0x7F) // double byte
+               {
+                   // in case we hit the NUL too early
+                   if (*p == '¥0')
+                   {
+                       break;
+                   }
+
+                   p++;
+               }
+               // else, single byte
+               // no-op
+           }
+       }
+       *vp = INT_TO_JSVAL(size);
+
+/* original SpiderMonkey code */
+#if 0
```

```
+       *vp = INT_TO_JSVAL((jsint) JSSTRING_LENGTH(str));
+#endif
+
+   }
     return JS_TRUE;
 }

-#define STRING_ELEMENT_ATTRS (JSPROP_ENUMERATE|JSPROP_READONLY|JSPROP_PERMANENT)
+// Bug-list HDTV-BML No00238 START
+//#define STRING_ELEMENT_ATTRS (JSPROP_ENUMERATE|JSPROP_READONLY|JSPROP_PERMANENT)
+#define STRING_ELEMENT_ATTRS (JSPROP_READONLY|JSPROP_PERMANENT)
+// Bug-list HDTV-BML No00238 END

 static JSBool
 str_enumerate(JSContext *cx, JSObject *obj)
@@ -714,8 +847,28 @@
             }
         }

-        str = js_NewDependentString(cx, str, (size_t)begin,
-                                    (size_t)(end - begin), 0);
+       /* problem
+            "漢字123".substring(0, 1)
+               should return "漢"
+               but now return 1st byte of "漢"
+
+          BugZilla #4055
+
+          so we need to adjust begin, end variables
+
+         --- 2007/03/28
+       */
+       {
+           size_t from =js_fix_FromEucCharIndexToCharIndex(str, (size_t)begin);
+           size_t to = js_fix_FromEucCharIndexToCharIndex(str, (size_t)end);
+           str = js_NewDependentString(cx, str, from, (to - from), 0);
+       }
+
+/* original SpiderMonkey code */
+#if 0
+       str = js_NewDependentString(cx, str, (size_t)begin,
+           (size_t)(end - begin), 0);
+#endif
         if (!str)
             return JS_FALSE;
     }
@@ -864,8 +1017,17 @@
     if (d < 0 || JSSTRING_LENGTH(str) <= d) {
         *rval = JS_GetEmptyStringValue(cx);
     } else {
-        index = (size_t)d;
-        str = js_NewDependentString(cx, str, index, 1, 0);
+       /* to support EUC, 2007/03/30
+        index = (size_t)d; */
+       /* one byte, or two bytes */
+       unsigned int one_or_two = 1;
+       index = js_fix_FromEucCharIndexToCharIndex(str, (size_t)d);
+       if (JSSTRING_CHARS(str)[index] > 0x7F)
+       {
+           one_or_two = 2;
+       }
+
+        str = js_NewDependentString(cx, str, index, one_or_two, 0);
         if (!str)
             return JS_FALSE;
         *rval = STRING_TO_JSVAL(str);
@@ -897,8 +1059,21 @@
     if (d < 0 || JSSTRING_LENGTH(str) <= d) {
         *rval = JS_GetNaNValue(cx);
     } else {
-        index = (size_t)d;
-        *rval = INT_TO_JSVAL((jsint) JSSTRING_CHARS(str)[index]);
+       /* to support EUC, 2007/03/30
+        index = (size_t)d;*/
+       const unsigned char * p = (unsigned char *)js_GetStringBytes(str);
+       index = js_fix_FromEucCharIndexToCharIndex(str, (size_t)d);
+       if (p[index] > 0x7F)
+       {
+           // two bytes
+           *rval = INT_TO_JSVAL((jsint)(p[index] << 8) | (jsint) p[index + 1]); // modified by yanjq 07/04/13
+       }
+       else
+       {
+           // one byte
+           *rval = INT_TO_JSVAL((jsint)p[index]);
+       }
+       //*rval = INT_TO_JSVAL((jsint) JSSTRING_CHARS(str)[index]);
     }
     return JS_TRUE;
 }
@@ -939,7 +1114,7 @@
 str_indexOf(JSContext *cx, JSObject *obj, uintN argc, jsval *argv, jsval *rval)
 {
     JSString *str, *str2;
-    jsint i, j, index, textlen, patlen;
+    jsint i, j, index, textlen, patlen, newindex;
     const jschar *text, *pat;
     jsdouble d;
```

```
@@ -975,6 +1150,14 @@
         return JS_TRUE;
     }

+//Bug-list HDTV-BML No00234 ADD-START
+    // インデックスの開始位置が全角文字に対して考慮されていないため、
+    // 終了時に行われている既存の半角/全角判定を行いインデックスを調整する処理をコールし、
+    // 開始位置についても全角に対応する。
+    newindex = js_fix_FromEucCharIndexToCharIndex( str, i );
+    i = newindex;
+//Bug-list HDTV-BML No00234 ADD-END
+
     /* XXX tune the BMH threshold (512) */
     if ((jsuint)(patlen - 2) <= BMH_PATLEN_MAX - 2 && textlen >= 512) {
         index = js_BoyerMooreHorspool(text, textlen, pat, patlen, i);
@@ -995,9 +1178,11 @@
             j = 0;
         }
     }
-
+
 out:
-    *rval = INT_TO_JSVAL(index);
+   // 2007/04/10
+   newindex =js_fix_FromCharCharIndexToEucIndex(str, index);
+    *rval = INT_TO_JSVAL(newindex);
     return JS_TRUE;
 }

@@ -1007,7 +1192,7 @@
 {
     JSString *str, *str2;
     const jschar *text, *pat;
-    jsint i, j, textlen, patlen;
+    jsint i, j, textlen, patlen, newindex;
     jsdouble d;

     str = js_ValueToString(cx, OBJECT_TO_JSVAL(obj));
@@ -1046,6 +1231,13 @@
         *rval = INT_TO_JSVAL(i);
         return JS_TRUE;
     }
+//Bug-list HDTV-BML No00234 ADD-START
+   // インデックスの開始位置が全角文字に対して考慮されていないため、
+   // 終了時に行われている既存の半角/全角判定を行いインデックスを調整する処理をコールし、
+   // 開始位置についても全角に対応する。
+   newindex = js_fix_FromEucCharIndexToCharIndex( str, i );
+   i = newindex;
+//Bug-list HDTV-BML No00234 ADD-END

     j = 0;
     while (i >= 0) {
@@ -1058,7 +1250,11 @@
             j = 0;
         }
     }
-    *rval = INT_TO_JSVAL(i);
+
+   // 2007/04/10
+   newindex =js_fix_FromCharCharIndexToEucIndex(str, i);
+
+    *rval = INT_TO_JSVAL(newindex);
     return JS_TRUE;
 }

@@ -1797,7 +1993,17 @@
             }
             return i + 1;
         }
-        return ((size_t)i == length) ? -1 : i + 1;
+
+//Bug-list HDTV-BML No00235 ADD-START
+//        return ((size_t)i == length) ? -1 : i + 1;
+        // セパレータ長が0の場合1文字ずつ切り出すが、全角の場合は、2文字分切り出す
+        if( chars[i] < 0x7F ) {
+            j = 1;      // 半角文字
+        } else {
+            j = 2;      // 全角文字
+        }
+        return ((size_t)i == length) ? -1 : i + j;
+//Bug-list HDTV-BML No00235 ADD-END
     }

     /*
@@ -2126,7 +2332,8 @@

     str = js_NewString(cx, tagbuf, taglen, 0);
     if (!str) {
-        free((char *)tagbuf);
+        //SafeFree((char *)tagbuf);/*2007-02-26*/
+       JS_PAGE_FREE((char *)tagbuf);// 2007/04/25
         return JS_FALSE;
     }
     *rval = STRING_TO_JSVAL(str);
@@ -2313,6 +2520,57 @@
     uint16 code;
     JSString *str;
```

```
+   /* for EUC support. need to split double bytes into single byte
+
+      e.g. for char code, 0xF7A1, we need to split it into 00F1, 00A1, 0000.
+
+    --- 2007/03/30
+   */
+   uintN new_argc;  // number of jschar needed
+   uintN new_i;
+
+    for (i = 0, new_argc = 0; i < argc; i++, new_argc++)
+   {
+        if (!js_ValueToUint16(cx, argv[i], &code))
+       {
+           return JS_FALSE;
+       }
+
+       if (code > 0xFF)
+       {
+           // double bytes
+           new_argc++;
+       }
+   }
+
+    chars = (jschar *) JS_malloc(cx, (new_argc + 1) * sizeof(jschar));
+    if (!chars)
+        return JS_FALSE;
+    for (i = 0, new_i = 0; i < argc; i++, new_i++) {
+        if (!js_ValueToUint16(cx, argv[i], &code)) {
+            JS_free(cx, chars);
+            return JS_FALSE;
+        }
+       if (code > 0xFF)
+       {
+           // double bytes
+           chars[new_i] = (jschar)code >> 8;
+           chars[++new_i] = (jschar)code  & 0xFF;
+       }
+       else
+       {
+// Bug-list HDTV-BML No00232 START
+//         chars[i] = (jschar)code;
+           chars[new_i] = (jschar)code;
+// Bug-list HDTV-BML No00232 END
+       }
+    }
+    chars[new_i] = 0;
+
+    str = js_NewString(cx, chars, new_argc, 0);
+
+/* original SpiderMonkey code */
+#if 0
     chars = (jschar *) JS_malloc(cx, (argc + 1) * sizeof(jschar));
     if (!chars)
         return JS_FALSE;
@@ -2325,6 +2583,9 @@
     }
     chars[i] = 0;
     str = js_NewString(cx, chars, argc, 0);
+
+#endif
+
     if (!str) {
         JS_free(cx, chars);
         return JS_FALSE;
@@ -2594,7 +2855,8 @@
 #ifdef DEBUG
         deflated_string_cache_bytes -= JSSTRING_LENGTH(str);
 #endif
-        free(he->value);
+        //SafeFree(he->value);/*2007-02-26*/
+       JS_PAGE_FREE(he->value);// 2007/04/25
         JS_HashTableRawRemove(deflated_string_cache, hep, he);
     }
     JS_RELEASE_LOCK(deflated_string_cache_lock);
@@ -2621,7 +2883,8 @@
         /* A stillborn string has null chars, so is not valid. */
         valid = (str->chars != NULL);
         if (valid)
-            free(str->chars);
+            //SafeFree(str->chars);/*2007-02-26*/
+           JS_PAGE_FREE(str->chars);// 2007/04/25
     }
     if (valid) {
         js_PurgeDeflatedStringCache(str);
@@ -2802,7 +3065,8 @@
     char *bytes;

     size = (length + 1) * sizeof(char);
-    bytes = (char *) (cx ? JS_malloc(cx, size) : malloc(size));
+    //bytes = (char *) (cx ? JS_malloc(cx, size) : SafeMalloc(size));/*2007-02-26*/
+   bytes = (char *) (cx ? JS_malloc(cx, size) : JS_PAGE_MALLOC(size));// 2007/04/25
     if (!bytes)
         return NULL;
     for (i = 0; i < length; i++)
@@ -2886,7 +3150,8 @@
                     deflated_string_cache_bytes += JSSTRING_LENGTH(str);
 #endif
```

```
                } else {
-                   free(bytes);
+                   //SafeFree(bytes);/*2007-02-26*/
+                  JS_PAGE_FREE(bytes);// 2007/04/25
                    bytes = NULL;
                }
            }
diff -u -d -r -N JS1.5ORG/jstypes.h JS1.5MOD/jstypes.h
--- JS1.5ORG/jstypes.h  2003-11-15 09:11:04.000000000 +0900
+++ JS1.5MOD/jstypes.h  2011-03-04 15:26:58.000000000 +0900
@@ -55,108 +55,12 @@
 #define jstypes_h___

 #include <stddef.h>
-
-/***********************************************************************
-** MACROS:      JS_EXTERN_API
-**              JS_EXPORT_API
-** DESCRIPTION:
-**      These are only for externally visible routines and globals.  For
-**      internal routines, just use "extern" for type checking and that
-**      will not export internal cross-file or forward-declared symbols.
-**      Define a macro for declaring procedures return types. We use this to
-**      deal with windoze specific type hackery for DLL definitions. Use
-**      JS_EXTERN_API when the prototype for the method is declared. Use
-**      JS_EXPORT_API for the implementation of the method.
-**
-** Example:
-**   in dowhim.h
-**     JS_EXTERN_API( void ) DoWhatIMean( void );
-**   in dowhim.c
-**     JS_EXPORT_API( void ) DoWhatIMean( void ) { return; }
-**
-**
-***********************************************************************/
-#ifdef WIN32
-/* These also work for __MWERKS__ */
-#define JS_EXTERN_API(__type) extern __declspec(dllexport) __type
-#define JS_EXPORT_API(__type) __declspec(dllexport) __type
-#define JS_EXTERN_DATA(__type) extern __declspec(dllexport) __type
-#define JS_EXPORT_DATA(__type) __declspec(dllexport) __type
-
-#define JS_DLL_CALLBACK
-#define JS_STATIC_DLL_CALLBACK(__x) static __x
-
-#elif defined(WIN16)
-
-#ifdef _WINDLL
-#define JS_EXTERN_API(__type) extern __type _cdecl _export _loadds
-#define JS_EXPORT_API(__type) __type _cdecl _export _loadds
-#define JS_EXTERN_DATA(__type) extern __type _export
-#define JS_EXPORT_DATA(__type) __type _export
-
-#define JS_DLL_CALLBACK             __cdecl __loadds
-#define JS_STATIC_DLL_CALLBACK(__x) static __x CALLBACK
-
-#else /* this must be .EXE */
-#define JS_EXTERN_API(__type) extern __type _cdecl _export
-#define JS_EXPORT_API(__type) __type _cdecl _export
-#define JS_EXTERN_DATA(__type) extern __type _export
-#define JS_EXPORT_DATA(__type) __type _export
-
-#define JS_DLL_CALLBACK             __cdecl __loadds
-#define JS_STATIC_DLL_CALLBACK(__x) __x JS_DLL_CALLBACK
-#endif /* _WINDLL */
-
-#elif defined(XP_MAC)
-#define JS_EXTERN_API(__type) extern __declspec(export) __type
-#define JS_EXPORT_API(__type) __declspec(export) __type
-#define JS_EXTERN_DATA(__type) extern __declspec(export) __type
-#define JS_EXPORT_DATA(__type) __declspec(export) __type
-
-#define JS_DLL_CALLBACK
-#define JS_STATIC_DLL_CALLBACK(__x) static __x
-
-#else /* Unix */
-
-#define JS_EXTERN_API(__type) extern __type
-#define JS_EXPORT_API(__type) __type
-#define JS_EXTERN_DATA(__type) extern __type
-#define JS_EXPORT_DATA(__type) __type
-
+/*2007-02-26*/
 #define JS_DLL_CALLBACK
 #define JS_STATIC_DLL_CALLBACK(__x) static __x

-#endif
-
-#ifdef _WIN32
-#  if defined(__MWERKS__) || defined(__GNUC__)
-#    define JS_IMPORT_API(__x)      __x
-#  else
-#    define JS_IMPORT_API(__x)      __declspec(dllimport) __x
-#  endif
-#else
-#    define JS_IMPORT_API(__x)      JS_EXPORT_API (__x)
-#endif
-
```

```
-#if defined(_WIN32) && !defined(__MWERKS__)
-#    define JS_IMPORT_DATA(__x)       __declspec(dllimport) __x
-#else
-#    define JS_IMPORT_DATA(__x)     __x
-#endif
-
-/*
- * The linkage of JS API functions differs depending on whether the file is
- * used within the JS library or not.  Any source file within the JS
- * interpreter should define EXPORT_JS_API whereas any client of the library
- * should not.
- */
-#ifdef EXPORT_JS_API
-#define JS_PUBLIC_API(t)    JS_EXPORT_API(t)
-#define JS_PUBLIC_DATA(t)   JS_EXPORT_DATA(t)
-#else
-#define JS_PUBLIC_API(t)    JS_IMPORT_API(t)
-#define JS_PUBLIC_DATA(t)   JS_IMPORT_DATA(t)
-#endif
+#define JS_PUBLIC_API(__type)    __type
+#define JS_PUBLIC_DATA(__type)   __type

 #define JS_FRIEND_API(t)    JS_PUBLIC_API(t)
 #define JS_FRIEND_DATA(t)   JS_PUBLIC_DATA(t)
@@ -214,8 +118,8 @@
 #define JS_ROUNDUP(x,y) (JS_HOWMANY(x,y)*(y))
 #define JS_MIN(x,y)     ((x)<(y)?(x):(y))
 #define JS_MAX(x,y)     ((x)>(y)?(x):(y))
-
-#if (defined(XP_MAC) || defined(XP_WIN)) && !defined(CROSS_COMPILE)
+/*2007-02-26*/
+#if (defined(XP_MAC) || defined(XP_WIN)) && !defined(CROSS_COMPILE) || defined(_ROSA)
 #    include "jscpucfg.h"        /* Use standard Mac or Windows configuration */
 #elif defined(XP_UNIX) || defined(XP_BEOS) || defined(XP_OS2) || defined(CROSS_COMPILE)
 #    include "jsautocfg.h"       /* Use auto-detected configuration */
@@ -223,7 +127,6 @@
 #else
 #    error "Must define one of XP_BEOS, XP_MAC, XP_OS2, XP_WIN or XP_UNIX"
 #endif
-
 JS_BEGIN_EXTERN_C

 /***********************************************************************
@@ -252,7 +155,6 @@
 #else
 #error No suitable type for JSInt16/JSUint16
 #endif
-
 /***********************************************************************
 ** TYPES:       JSUint32
 **              JSInt32
@@ -323,7 +225,6 @@
 #else
 #error 'sizeof(int)' not sufficient for platform use
 #endif
-
 /***********************************************************************
 ** TYPES:       JSFloat64
 ** DESCRIPTION:
diff -u -d -r -N JS1.5ORG/jsutil.c JS1.5MOD/jsutil.c
--- JS1.5ORG/jsutil.c   2004-02-25 22:33:42.000000000 +0900
+++ JS1.5MOD/jsutil.c   2011-03-04 15:21:54.000000000 +0900
@@ -42,7 +42,8 @@
  * PR assertion checker.
  */
 #include "jsstddef.h"
-#include <stdio.h>
+/*#include <stdio.h> 2007-02-26*/
+#include "ebml_fio.h"
 #include <stdlib.h>
 #include "jstypes.h"
 #include "jsutil.h"
@@ -140,18 +141,19 @@

 JS_PUBLIC_API(void) JS_Assert(const char *s, const char *file, JSIntn ln)
 {
-#ifdef XP_MAC
-    dprintf("Assertion failure: %s, at %s:%d¥n", s, file, ln);
-#else
-    fprintf(stderr, "Assertion failure: %s, at %s:%d¥n", s, file, ln);
-#endif
+/*2007-02-26*/
+//#if !defined (_ENVY_PORTING_)
+//#error Envy移植：要確認
+//    fprintf(stderr, "Assertion failure: %s, at %s:%d¥n", s, file, ln);
+//#endif
 #if defined(WIN32)
     DebugBreak();
+    exit(3);
 #endif
 #if defined(XP_OS2)
     asm("int $3");
 #endif
-#ifndef XP_MAC
+#if !defined(_ROSA)/*2007-02-26*/
     abort();
 #endif
 }
```

```
diff -u -d -r -N JS1.5ORG/jsutil.h JS1.5MOD/jsutil.h
--- JS1.5ORG/jsutil.h   2004-02-25 23:05:28.000000000 +0900
+++ JS1.5MOD/jsutil.h   2011-03-04 15:27:10.000000000 +0900
@@ -44,6 +44,7 @@
 #ifndef jsutil_h___
 #define jsutil_h___

+#include "bmlmem.h"/*2007-02-26*/
 JS_BEGIN_EXTERN_C

 #ifdef DEBUG
diff -u -d -r -N JS1.5ORG/prmjtime.c JS1.5MOD/prmjtime.c
--- JS1.5ORG/prmjtime.c 2004-08-04 11:28:58.000000000 +0900
+++ JS1.5MOD/prmjtime.c 2011-03-04 16:36:38.000000000 +0900
@@ -45,7 +45,10 @@
 #define _REENTRANT 1
 #endif
 #include <string.h>
+#ifdef _ROSA/*2007-02-26*/
+#else
 #include <time.h>
+#endif
 #include "jstypes.h"
 #include "jsutil.h"

@@ -58,8 +61,9 @@
 #include <sys/timeb.h>
 #endif
 #ifdef XP_WIN
-#include <windef.h>
-#include <winbase.h>
+/*#include <windef.h>
+#include <winbase.h> *//*2007-02-26*/
+#include <windows.h>
 #endif

 #ifdef XP_MAC
@@ -265,6 +269,13 @@
         gmtOffsetSeconds -= PRMJ_HOUR_SECONDS;
     return (zone = -gmtOffsetSeconds);
 #endif
+
+#ifdef _ROSA /*20070409*/
+   /*LocalTZA of Date object must be fixed to +9 hours of Japan Standard Time.*/
+   /*for we cann't get TIMEZONE by system interface,we just set const value here*/
+   return (-9) * 3600;
+#endif
+
 }

 /* Constants for GMT offset from 1970 */
@@ -396,31 +407,23 @@

     return *((JSUint64 *)&localTime);
 #endif /* XP_MAC */
+#ifdef _ROSA /*20070409*/
+{
+   extern JSInt64 rosa_prmj_now();
+   return rosa_prmj_now();
+}
+#endif /*_ROSA*/
 }

 /* Get the DST timezone offset for the time passed in */
 JSInt64
 PRMJ_DSTOffset(JSInt64 local_time)
 {
-    JSInt64 us2s;
-#ifdef XP_MAC
-    /*
-     * Convert the local time passed in to Macintosh epoch seconds. Use UTC utilities to convert
-     * to UTC time, then compare difference with our GMT offset. If they are the same, then
-     * DST must not be in effect for the input date/time.
-     */
-    UInt32 macLocalSeconds = (local_time / PRMJ_USEC_PER_SEC) + gJanuaryFirst1970Seconds, utcSeconds;
-    ConvertLocalTimeToUTC(macLocalSeconds, &utcSeconds);
-    if ((utcSeconds - macLocalSeconds) == PRMJ_LocalGMTDifference())
-        return 0;
-    else {
-        JSInt64 dlsOffset;
-       JSLL_UI2L(us2s, PRMJ_USEC_PER_SEC);
-       JSLL_UI2L(dlsOffset, PRMJ_HOUR_SECONDS);
-       JSLL_MUL(dlsOffset, dlsOffset, us2s);
-        return dlsOffset;
-    }
+#if defined(_ROSA)/*2007-02-26*/
+   JSInt64 tmp = {0};
+   return tmp;
 #else
+    JSInt64 us2s;
     time_t local;
     JSInt32 diff;
     JSInt64  maxtimet;
@@ -474,6 +477,9 @@
 size_t
 PRMJ_FormatTime(char *buf, int buflen, char *fmt, PRMJTime *prtm)
 {
+// Bug-list HDTV-BML #00297 ADD-START
```

```
+   size_t result = 0;
+// Bug-list HDTV-BML #00297 ADD-END
 #if defined(XP_UNIX) || defined(XP_WIN) || defined(XP_OS2) || defined(XP_MAC) || defined(XP_BEOS)
     struct tm a;

@@ -523,6 +529,9 @@

     return strftime(buf, buflen, fmt, &a);
 #endif
+// Bug-list HDTV-BML #00297 ADD-START
+   return result;
+// Bug-list HDTV-BML #00297 ADD-END
 }

 /* table for number of days in a month */
diff -u -d -r -N JS1.5ORG/prmjtime.h JS1.5MOD/prmjtime.h
--- JS1.5ORG/prmjtime.h 2003-11-15 09:11:04.000000000 +0900
+++ JS1.5MOD/prmjtime.h 2011-03-04 16:42:32.000000000 +0900
@@ -43,7 +43,13 @@
  * PR date stuff for mocha and java. Placed here temporarily not to break
  * Navigator and localize changes to mocha.
  */
+ /* 2007.02.26 */
+#ifdef _ROSA
+typedef long time_t;
+#else
 #include <time.h>
+#endif
+
 #include "jslong.h"
 #ifdef MOZILLA_CLIENT
 #include "jscompat.h"
@@ -57,21 +63,21 @@
  * Broken down form of 64 bit time value.
  */
 struct PRMJTime {
-    JSInt32 tm_usec;       /* microseconds of second (0-999999) */
-    JSInt8 tm_sec;     /* seconds of minute (0-59) */
-    JSInt8 tm_min;     /* minutes of hour (0-59) */
-    JSInt8 tm_hour;        /* hour of day (0-23) */
-    JSInt8 tm_mday;        /* day of month (1-31) */
-    JSInt8 tm_mon;     /* month of year (0-11) */
-    JSInt8 tm_wday;        /* 0=sunday, 1=monday, ... */
-    JSInt16 tm_year;       /* absolute year, AD */
-    JSInt16 tm_yday;       /* day of year (0 to 365) */
-    JSInt8 tm_isdst;       /* non-zero if DST in effect */
+    JSInt32 tm_usec;            /* microseconds of second (0-999999) */
+    JSInt8 tm_sec;              /* seconds of minute (0-59) */
+    JSInt8 tm_min;              /* minutes of hour (0-59) */
+    JSInt8 tm_hour;             /* hour of day (0-23) */
+    JSInt8 tm_mday;             /* day of month (1-31) */
+    JSInt8 tm_mon;              /* month of year (0-11) */
+    JSInt8 tm_wday;             /* 0=sunday, 1=monday, ... */
+    JSInt16 tm_year;            /* absolute year, AD */
+    JSInt16 tm_yday;            /* day of year (0 to 365) */
+    JSInt8 tm_isdst;            /* non-zero if DST in effect */
 };

 /* Some handy constants */
-#define PRMJ_USEC_PER_SEC  1000000L
-#define PRMJ_USEC_PER_MSEC 1000L
+#define PRMJ_USEC_PER_SEC       1000000L
+#define PRMJ_USEC_PER_MSEC      1000L

 /* Return the current local time in micro-seconds */
 extern JSInt64
```