Mavimax, Ltd

Enduro/X Middleware Data Sheet

March 2018

EXECUTIVE SUMMARY

Enduro/X is an open source middleware platform for distributed transaction processing. It is built on proven APIs such as X/Open group's XATMI and XA. Platform is designed for building realtime microservices based applications with clusterization option. Enduro/X functions as extended drop-in replacement alternative for Oracle Tuxedo. The platform uses in-memory POSIX Kernel queues which insures high interprocess communication throughput.

This document lists key characteristics of the Enduro/X middleware platform.

PARAMETERS

This section lists various aspects of the Enduro/X middleware platform. Starting for very general parameters and ending with very specific platform describing attributes.

General middleware description	
Supported operating systems	Linux, IBM AIX, Oracle Solaris, FreeBSD, MacOS X
Internal IPC mechanisms	Posix Kernel Queues (where possible), Posix shared memory and System V Semaphores. In fall-back mode queues are simulated within shared memory with help pthread shared mutexes.
Application server	One of the Enduro/X functionalities are general purpose application server which monitors XATMI servers and clients. Clients can be non related to XATMI, in that case Enduro/X will monitor their life-cycle (auto-boot, start, stop, reboot faulty processes).
Systems model	Open
Standards Supported	SCA, X/Open group's XATMI and XA
Class A APIs	C/C++, Go
Class B APIs (community based)	Perl, PHP
Class C APIs (community based)	Python, Node.js

Enduro/X provides application server bridging in network. This gives ability to connect multiple processing machines in single application cluster. The internal XATMI server and clients have an abstraction of the services within the cluster. The actual involved software components in data processing are not aware of the physical node which executes the request. This gives developers great advantage of thinking about services and not about physical connectivity. The connectivity later will be configured by administrators.

Clustering options - bridge	
Cluster mode	Peer-to-Peer, no master node.
Mechanism	Special XATMI Server “tpbridge”. Monitored by local Enduro/X daemon instance as standard XATMI server.
Network	TCP/IP Sockets
Connection roles	Each “tpbridge” can be configured to be as TCP Client or TCP Server.
Security	Optional RSA traffic encryption with GPG
Layer 7 protocol	Enduro/X provides two operating modes of the “tpbridge”: 1. Platform specific, optimal for same architecture type machine interconnection; 2. Platform agnostic, TLV protocol for connecting different type of processing machines;
Max number of nodes	Currently 32, but can be extended till 255.
Optimizations	When processing responses in within cluster, Enduro/X tries to find the shortest route to deliver response to caller.

Enduro/X provides clustering options for HTTP/REST services, so that XATMI application can be simply integrated with external web services.

Clustering options – HTTP/REST	
Outgoing REST connects	Enduro/X Connect package provides special XATMI server, named “restoutsv” which can map th external HTTP(S)/REST services as a standard XATMI services. Different buffer formats are supported. The external services can be monitored via pings and system can expose only those services which are live, unavailable services are automatically unadvertised from the XATMI sub-system.
Incoming REST connects	Enduro/X Connect package allows to expose XATMI services as HTTP(S)/REST resources. This functionality provided by “restincl” XATMI client process. Different buffer formats can be used. Service mapping is done with extensive set of configuration options.

XATMI stands for Application-to-Transaction Monitor Interface. Thus at the core of the API, system is designed for transaction processing. Enduro/X supports two phase XA transactions.

Distributed transaction processing support	
Protocol	X/Open XA, two phase commit
Transaction manager	Included transaction manager. Works on peer to peer basis. The cluster

Distributed transaction processing support	
	node which initiated transaction, becomes “master” for particular transaction, and it completes the transaction till the finish i.e. commit or abort.
Logging	Transactions are logged at prepare stage by transaction manager. At system reboots, transaction manager continues to finish open transactions.
Load balancing	Enduro/X transaction manager architecture is build on principles that they can be load balanced and multiple instances can be started.
Resource managers supported out of the box	1. Any driver which provides standard Open/X XA API. Currently drivers are provided: A) IBM DB2. B) WebSphere Message Queue. C) Oracle DB. D) Other XA Compliant driver is easy to adopt for Enduro/X, needs a wrapper library which provides XA Switch symbol to Enduro/X. 2. Oracle DB driver for Go - “go-oci8”, provided by Mavimax.

The XATMI API supports different types of Inter Process Communication mechanisms. Which might be simple synchronous and asynchronous request-reply, session based conversational and eventing, so that processes can broadcast the data to interested subscribers. Enduro/X provides different set of protocol buffers that can be used within data transport.

Messaging protocols	
Synchronous	Most basic services invocation with processes waiting for answer. This is basic “tpcall()” invocation.
Asynchronous	Process can invoke multiple services and can wait for their completion asynchronously. This is realized with “tpacall()” and “tpgetrply()”.
Conversational	In conversational mode process (XATMI client or server) can invoke other XATMI server in conversational mode. Communications are semi-bidirectional. API: “tpconnect()”, “tprecv()”, “tpsend()”, “tpdiscon()”.
Publish-and-subscribe	Event brokering provides feature for XATMI server to subscribe for some specific event name (direct keyword or regex matched). When XATMI client or server publishes the event via “tppost()”, the corresponding servers will receive the message. Subscriptions are done via “tpsubscribe()” and “tpunsubscribe()”.
Unsolicited	Enduro/X supports unsolicited messages. Meaning that during the standard service call, asynchronous messages can be delivered back to original

Messaging protocols	
	caller. The original call is not interrupted. The same way multiple client processes can be notified via broadcast mechanism.
Timeout control	Each call to service is monitored by globally set timeout parameter. If the timeout is reached, the invocation call will return corresponding error code. If needed, timeout can be obeyed . If timeout is used for particular call and it expires, the Enduro/X middleware will drop the message within the call session, if expired message is about to be forwarded to another service.

For doing message exchange between XATMI clients and XATMI servers, the message must conform to some logical format. The format provided by Enduro/X is API to allocate specific type of the buffers. For each of the buffer type there is special API.

IPC Buffers	
Unified Buffer Format (UBF)	This is protocol buffer. Very similar to Oracle Tuxedo's FML. Enduro/X even provides emulation of the FML, via "fml.h" and "fml32.h". The buffer contains arbitrary number typed (char, short, float, double, string and raw/carray) named fields. Each field might have a number of occurrences. Basically it is key-value hash of the arrays. The keys are mostly indexed and data is kept in linear memory. The processing is the data buffer is very light and provides good performance when comparing to JSON protocol buffers of the other languages. The format is preferred format for large scale application development. As the concept of the key-value buffer forwards and backwards compatibility between new and existing XATMI binaries. Enduro/X provides highly extensive API for UBF protocol buffer processing. It even includes script evaluations on the buffer. The API contains about 60 different calls for UBF.
STRING	This is basic string buffer, Enduro/X can be loaded with arbitrary zero terminated string data for sending to services and reading the value back with standard C string handler. For high level languages API is provided to manipulate the buffer.
JSON	This is JSON string loaded into XATMI allocated buffer. There is API calls for converting the UBF to JSON and vice versa.
CARRAY/X_OCTET	This raw data buffer (can contain 0x00 byte). Enduro/X provides standard API to read and write values to. For C it is standard memory pointer.
VIEW	This feature is added to Enduro/X for more compatibility with Oracle Tuxedo. View allows programmers to defined view files. View file contains structure filed descriptions and various options, like mapped UBF buffer fields. VIEW allows to transfer records over the different platforms in cross platform way. And VIEW buffers makes it easier to operate with

IPC Buffers	
	UBF buffers, offering more streamline approach.
Buffer management	Enduro/X tracks the memory allocations and at certain points provides garbage collection of the automatic buffers. For high level languages where destructor are available, for example Go, the full buffer deallocation is done automatically.

Enduro/X provides persistent queue interfaces. The messages are persisted to disk by queue resource manager and can be dequeued automatically (messages are sent to configured target XATMI server)

Persistent queues	
Modes	Store -and-Forward (SAF), manual enqueue dequeue
Configuration	Manual and automatic (new queues can be based on standard queue presets)
Transactions	Enduro/X persisted queue sub-system is exported as XA Resource Manager. Thus if running in global transaction mode, “tpenqueue()" will invoke message queue in transactional mode and will be do commit or abort together with other resource manager transactions, e.g. database.
Load balance	Multiple queues spaces can be served by different queue daemons. If using only SAF mode, then in singe space multiple daemons can be booted. The Queue daemon is special XATMI server, but booted as standard service.
Data format	Each message is enqueued in file system as a binary file. It is possible to transfer files from one machine to another, if for some reason machine becomes broken and unavailable.
API	“tpenqueue()”, “tpdequeue()”

Enduro/X was build with performance in mind. The core of Enduro/X written in C which provides very effective operating system resource usage. Following table list the Enduro/X benchmark attributes on commodity desktop hardware. This makes some insight of the performance on production hardware of the multiple processor cores and lot of memory.

Performance - IPC @i7 6600U, ssd, 16GB Ram, Linux kernel 4.4	
Synchronous service call, one client, one server IPC (request/reply)	~118,000 calls/sec
Asynchronous one client calls one server (request only)	~500,000 calls/sec

Synchronous service call, 5x clients calls 5x servers (request/reply)	~190,000 calls/sec
Bridged (two cluster nodes connected on localhost), client on one cluster node calls server on another cluster node (request/reply)	~25,000 calls/sec
Persistent storage, messages enqueued to solid storage	~2100 msgs/sec
Tpcall() cached results	~800'000 request/reply calls per second @ 1 KB buffer

Each developed application must be protected from unauthorized access. Enduro/X provides several options for cluster application security.

Security	
Basic security model	Enduro/X uses operating system kernel queues. Queues as a other OS resources are protected by user and group access. Thus the sample Unix security principle applies to Enduro/X middleware.
Cluster bridge	Cluster bridge can be secured by GPG RSA or DSA asymmetrical PKI infrastructure. It is possible to configure cluster link to encrypt and signing with local node's private key and signing with target nodes public key. Thus gaining high extent of the traffic encryption.
Bridge operations	The service exported to other cluster nodes can be blacklisted or white-listed on each bridge basis, so that other cluster node "sees" only certain services.
REST/HTTP SSL	Enduro/X Connect provides SSL traffic encryption for incoming and output service mapping operations.
REST/HTTP visibility	Service which are available from Enduro/X Connect package are defined at configuration level. Thus external REST caller cannot access unexported services.

The platform is fundamentally built on technologies and principles which allow to scale platform linearly and load balance with simplicity. This is reason why kernel queues were select for platform as these atoms provides very efficient inter-process communications framework. In contrast with TCP/IP sockets, kernel queues requires far less processing as network stack is quite complicated. Also TCP/IP sockets are streaming protocol, however kernel queues are atomic messages with guaranteed single read full message delivery.

Scalability and Reliability	
Application parallelization	Enduro/X allows applications to handle requests in parallel and process

Scalability and Reliability	
	multiple transactions simultaneously on different, distributed nodes, thus eliminating single point of failure and increasing scalability
Linear Scalability	Endruo/X provides almost linear increase in application throughput corresponding to increase in available resources. New resources can be plugged in existing cluster and it load will be balanced
Local load balancer approach	• On operation system where available, One queue – multiple servers approach is used. This means that any first free XATMI server will take next request for processing. This option is available on Linux and FreeBSD platforms. • For other platforms (AIX, Solaris, MacOS) round-robin approach is used for dispatching requests accessors the same XATMI service providers.
Cluster operations load balancer approach	When service is available in local node and remote node, the global variable can be set to redirect certain amount of requests on remote services. This can be controlled in percentage. And percentage is calculated on random basis. When service is available only in cluster, then they are invoked directly no matter of the percentage settings.
Caching	Enduro/X Smart Cache provides flexible way to configure distributed service call caching. Once configuration is done, the first service call is performed for “real” and then results are written to cache. The next service calls are returned directly from memory mapped file, which usually is kept in RAM by operating system. As a results this greatly increases service call performance. The configuration includes different rules like when to cache, what data to cache, how long data may stay in cache, how to refresh the caches. Cluster operations are supported - once call is cached on some cluster node, the cache data may be replicated to other nodes.
Replicated service framework	Administrators can dynamically replicates services across the network to maximize performance and reliability.
Failover	Enduro/X automatically redirects all the traffic to alive services meanwhile removing failed components from cluster visibility.
Robust fault management	It is possible to minimizes downtime and keep applications running through planned and unplanned downtime by eliminating single points of failure. The cluster nodes can be connected and disconnect in real-time and system will be automatically reconfigured for new operations.
Service monitoring	On each local cluster node, XATMI servers are monitored by Enduro/X local daemon instance. The daemon performs periodic process pings. When process does not respond for several pings (configured number of

Scalability and Reliability	
	times), the process gets gracefully shutdown or killed if does not respond at all. The process resources are moved from system and new instance is booted. Thus this feature is part of the system self-healing.
Core fault recovery	As part of the self-healing approach, it is even possible to reboot the local Enduro/X daemon with out affecting the core system processes. The XATMI servers and clients can work dependently from the daemon. When daemon process (“ndrxd”) is started back, it is put in learning mode, it grabs the infos from the system and after a while becomes local master again. This process can be made automatic by special XATMI server named “tprecover”. When this service is booted, it monitors “ndrxd” and “ndrxd” monitors “tprecover” thus there is no single point of failure.
Real-time system patching	The services in Enduro/X application cluster can be replaced in real time with out system stopping or reboot. This can be done due to fact that typically services are stateless and their calls are automatically load balanced. Thus if doing graceful shutdown and start back with new XATMI server process, causes zero loss of transactions and no service interruption.
System sanity	Enduro/X main local cluster node daemon is following up the system status and periodically tests all the resources allocated on the IPC sub-system. If some resources are found to be belonging to non-existing process, the resources are automatically removed. In the same way, the daemon is following up the configuration and reboots any missing processes, if they have failed and exited.
Logging	Enduro/X is built on solid debug logging framework. Each major component logs the processing details at different log levels. The logs can be redirected on process bases to separate files. Log levels can be controlled for XATMI and UBF facilities. This gives developers great chance to look in the internals of the Enduro/X when resolving some problem.
Error reporting	Enduro/X provides additional diagnostics within the standard XATMI framework. For example “tpstrerror()” will not only provide format message of the error occurred, but it will also contain additional message from what exactly occurred and why. This feature is very helpful when resolve some hard to understand problem.

Some of the Enduro/X internal library features are seen to be good for general purpose use. Thus Mavimax exports this functionality and provides it with the standard Enduro/X API.

Infrastructure	
Common logging framework	Enduro/X provides formal logging API named “tplog()”, “tplogdump()”

Infrastructure	
	and “tplogdumpdiff()”, however at C level header “ndebug.h” is available for even better logging with C macros which will include exact source file and code line which generated the log entry. This framework is available for high level programming languages like Go. Thus when developing new applications, developers are no need to seek for good logging library. It comes out of the box with Enduro/X.
Common configuration framework	Another significant feature of the applications is configuration storage and access. Enduro/X provides highly scalable access to INI files. It is matter of standard service call for particular process to receive a UBF buffer with key-value configuration details for specified INI file subsection. The Common Configuration framework can use different resources for INI, for example exact files or directories with files. All they are merged, subsections resolved and merged data is provided to process which is requesting configuration. The configuration is cached and provides fast access.
Runtime provision	When it comes to system deployment, Enduro/X provides functionality for quick runtime system preparation. This can be done by help Enduro/X command line administration tool “xadmin”. The “xadmin provision” will ask several questions with defaults offered and in the end will generate best practices based runtime configuration.
Built in code generators	Enduro/X ships with build in generators for UBF tables, Go and C XATMI server and clients. The available targets can be found by “xadmin help gen”. These are shell based wizards which in the end will generate basic process sources with which developers can start to work to implement business logic needed.
API for external code generators	The generators framework is user extensible and additional wizards based targets can be plugged in “xadmin” CLI utility. Thus the vendor company can initially write a set of generators, so that later it would be simpler to develop internal standards based applications.

As Mavimax Enduro/X implements XATMI standard, it provides some sort of migration options from other transaction monitors. The following table marks the notes for the compatibility.

Compatiblity	
Oracle Tuxedo	Enduro/X is positioned as Oracle Tuxedo alternative. The compatibility grows with each of the Enduro/X major release. Currently it is estimated %80. For example, there are few functions written as stubs, missing “view” support (soon will be implemented), missing X/Open TX interface (similar transaction operations are provided in XATMI).
RedHat Blacktie	Enduro/X compatibility with Blacktie is about 95%.

Hitache OpenTP1	Enduro/X compatibility with OpenTP1 in segment of XATMI is about 97%.

System on chip silicons become more and more available and the systems need to be managed. Enduro/X as a lightweight middleware can be deployed in small system chips to make and abstraction of Internet of Things (IoT) sensors to be orchestrated as XATMI services. Enduro/X is moving towards IoT and the table bellow lists current status.

IoT Compatbility	
Platforms	Raspberry PI are supported out of the box, Other small systems which can run the Linux, can be supported too.
TCP	Enduro/X Connect package provides generic TCP driver which abstracts sockets as XATMI services. This function can be used for IoT when driving custom protocol based TCP/IP sensors and chips, for example esp8266.

There are certain limits for the Enduro/X platform. The following table lists them.

Limitations	
Forbidden symbols in executables and service names	“,”, Service names cannot start with “@”.
Maximum identifier length (service named, UBF field name)	30 ASCII characters.
Maximum cluster nodes	32 out of the box, 255 hypothetical limit
Maximum IPC buffer size	64 KB. Starting with Enduro/X 5.1 buffer size is limited to stack size (several megabytes or even more depending on system).
Maximums simultaneous asynchronous service calls by one thread	Normally 16384. On MacOS 1000
Maximum simultaneous conversations calls from one thread	5
Maximum executable name	~60 ASCII characters
Maximum number of resource managers (different databases) participating in cluster app with single transaction	32

API OVERVIEW

This section list the API support for particular programming languages. The API is grouped in functional blocks. The cells within the table gives the idea of API compliance with full XATMI functionality. Legend is following:

- "+++" - Fully featured API call.
- "++" - Partially implementation.
- "+" - Very basic call functionality.
- space – not available.

XATMI API Calls

XATMI calls are intended for doing inter process communications and transaction life-cycle management. This table lists the XATMI API cross language compatibility.

XATMI Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
Transport/IPC						
tpcall	+++	+++	++	++	++	+
tpacall	+++	+++	++	++	++	
tpconnect	+++	+++		++	++	
tpgetrply	+++	+++	++	++	++	
tprecv	+++	+++		++	++	
tpsend	+++	+++		++	++	
tpreturn	+++	+++		++	++	
tpforward	+++	+++		++		
tpunadvertise	+++	+++		++		
tpunsubscribe	+++	+++		++	+	
tppost	+++	+++		++	+	
tpdequeue	+++	+++		++	+	
tpenqueue	+++	+++		++	+	
tpdiscon	+++	+++		+++	+++	
tpcancel	+++	+++			+++	
tpadvertise	+++	+++		+++	+++	
tpsvrdone	+++	+++		+++	+++	
tpsvrinit	+++	+++		+++	+++	
tpsubscribe	+++	+++		++	++	
tpservice	+++	+++		++	++	
tpcontinue	+++	+++				
tpnotify	+++					
tpbroadcast	+++					

XATMI Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
tpchkunsol	+++					
tpsetunsol	+++					
Transactions						
tpbegin	+++	+++	+++	+++	+++	
tpresume	+++	+++		+++	++	
tpopen	+++	+++		+++	+++	
tpclose	+++	+++	++	+++	+++	
tpcommit	+++	+++	++	+++	+++	
tpabort	+++	+++	++	+++	+++	
tpgetlev	+++	+++		+++	+++	
tpsuspend	+++	+++		++	++	
Generic Buffer Management						
tpalloc	+++	+++	+++	++	++	
tpfree	+++	+++	+++	+	++	
tpisautobuf	+++	+++				
tpjsontoubf	+++	+++				
tpubftojson	+++	+++				
tptypes	+++	+++		++	+++	
tprealloc	+++	+++	+++		+++	
Extensions & hooks						
tpext_addb4pollcb	+++	+++				
tpext_addperiodcb	+++	+++				
tpext_addpollerfd	+++	+++				
tpext_delb4pollcb	+++	+++				
tpext_delperiodcb	+++	+++				
tpext_delpollerfd	+++	+++				
Multithreading						
tpgetctxt	+++	embedded		+		
tpnewctxt	+++					
tpfreectxt	+++					
tpsetctxt	+++			+		
tpsrvgetctxdata	+++	+++				
tpsrvsetctxdata	+++	+++				
Logging						
tplog	+++	+++		+++		

XATMI Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
tplogclosereqfile	+++	+++				
tplogclosethread	+++	+++				
tplogconfig	+++	+++				
tplogdelbufreqfile	+++	+++				
tplogdump	+++	+++				
tplogdumpdiff	+++	+++				
tploggetbufreqfile	+++	+++				
tploggetreqfile	+++	+++				
tplogprintubf	+++	+++				
tplogsetreqfile	+++	+++				
tplogsetreqfile_direct	+++	+++				
Auxiliary & integration						
ndrx_main	+++					
ndrx_main_integra	+++					
tpgetnodeid	+++	+++		+++		
tpgetsrvid	+++	+++				
tpinit	+++	+++	+++	++	++	
tpstrerror	+++	+++	+++	++	++	
tpterm	+++	+++	+++	+++	++	

UBF API Calls

This section lists Unfied Buffer Formats (UBF) protocol buffer API calls and their compatibility with other programming languages.

UBF Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
Field management						
Badd	+++	+++	++	+ (ubf2dict)	+	+ (ubf2json)
Badds	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bchg	+++	+++	+++ (Badd)	+ (ubf2dict)		+ (ubf2json)
Bchgs	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bdel	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bfind	+++	++ (Bget)	arrayubf	+ (ubf2dict)		+ (ubf2json)
Bfindlast	+++			+ (ubf2dict)		+ (ubf2json)
Bfindocc	+++	++ (Bget)		+ (ubf2dict)		+ (ubf2json)

UBF Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
Bfinds	+++	++ (Bget)		+ (ubf2dict)		+ (ubf2json)
Bget	+++	+++	+++	+ (ubf2dict)	++	+ (ubf2json)
Bgetalloc	+++			+ (ubf2dict)		+ (ubf2json)
Bgetlast	+++			+ (ubf2dict)		+ (ubf2json)
Bgetsa	+++			+ (ubf2dict)		+ (ubf2json)
Bgets	+++			+ (ubf2dict)		+ (ubf2json)
Blen	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Boccur	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bpres	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
CBadd	+++	+++ (Badd)		+ (ubf2dict)		+ (ubf2json)
CBchg	+++	+++ (Bchg)		+ (ubf2dict)		+ (ubf2json)
CBfind	+++			+ (ubf2dict)		+ (ubf2json)
CBfindocc	+++			+ (ubf2dict)		+ (ubf2json)
CBget	+++	+++ (Bget)	+++ (Bget)	+ (ubf2dict)		+ (ubf2json)
CBgetalloc	+++	+++ (Bget)	+++ (Bget)	+ (ubf2dict)		+ (ubf2json)
Field type management						
Bfldid	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bfldno	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bfldtype	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bfname	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bmkfldid	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Btypcvt	+++	+++		+ (ubf2dict)		+ (ubf2json)
Btype	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Buffer management						
Balloc	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bconcat	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bcpy	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bdelall	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bdelete	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bextread	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bfprint	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bfree	+++			+ (ubf2dict)		+ (ubf2json)
Bnext	+++	+++	+(arrayubf)	+ (ubf2dict)		+ (ubf2json)
Bprint	+++	+++		+ (ubf2dict)	++	+ (ubf2json)
Bproj	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bprojcpy	+++	+++		+ (ubf2dict)		+ (ubf2json)

UBF Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
Bread	+++	+++		+ (ubf2dict)		+ (ubf2json)
Brealloc	+++			+ (ubf2dict)		+ (ubf2json)
Bsizeof	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bunused	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bupdate	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bused	+++	+++	+++	+ (ubf2dict)		+ (ubf2json)
Bwrite	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bcmp	+++					
VIEW Buffer management						
Bvftos	+++					
Bvstof	+++					
Bvnull	+++					
Bvopt	+++					
Bvselinit	+++					
Bvsinit	+++					
CBvget	+++	+++				
CBvchg	+++	+++				
Bvsizeof	+++	+++				
Bvcpy	+++	+++				
Bvoccur	+++	+++				
Bvsetoccur	+++	+++				
Bvnext	+++	+++				
Expression & scripting						
Bboolco	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bboolev	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bboolpr	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bboolsetcbf	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bfloatev	+++	+++		+ (ubf2dict)		+ (ubf2json)
Btreefree	+++	+++		+ (ubf2dict)		+ (ubf2json)
Auxiliary						
Berror	+++	+++	+++	+ (ubf2dict)	++	+ (ubf2json)
Bidxused (stub)	+		+			+ (ubf2json)
Bindex (stub)	+				+	+ (ubf2json)
Brstrindex (stub)	+		+			+ (ubf2json)
Binit	+++	+++		+ (ubf2dict)		+ (ubf2json)
Bisubf	+++	+++	+			+ (ubf2json)

UBF Call	C/C++	Go	PHP	Python	Perl	Node.Js (Javascript)
Bstrerror	+++	+++	+	+ (ubf2dict)	++	+ (ubf2json)
Bunindex (stub)	+		+			+ (ubf2json)