



Mavimax, Ltd

Enduro/X Middleware Data Sheet

August 2022

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
CPU Architectures	X86, X86_64, PowerPC, SPARC, ARM
Internal IPC mechanisms	Posix Kernel Queues (where possible), System V Queues, System V shared memory and System V Semaphores. In fall-back mode queues are emulated 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
APIs	C/C++, Go, Java, Python
Community based APIs	Perl, PHP

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.
Layer 7 protocol	Enduro/X provides two operating modes of the “tpbridge”: <ol style="list-style-type: none"> 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, including generic format where http protocol fields are loaded into UBF buffer for request and response. 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.
Automatic transactions	Services can be configured to be auto-transactional. If such service is invoked outside the global transaction, Enduro/X will managed the transaction life cycle for the invoked chain of services.
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	<ol style="list-style-type: none"> 1. Any driver which provides standard Open/X XA API. Currently drivers are provided: <ol style="list-style-type: none"> A) IBM DB2. B) WebSphere Message Queue. C) Oracle DB. D) PosgreSQL. E) 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. 3. JDBC compatible resources (depends on database).

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 “tpgetreply()”.
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

Messaging protocols	
	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 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 (R) Tuxedo’s (R) 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 120 different calls for UBF. UBF buffer API supports recursive field manipulation. As well support for PTR and VIEW types are provided too.
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.

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

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.x, Enduro/X version 7.5	
Synchronous service call, one client, one server IPC (request/reply)	~72,000 calls/sec @ 1 KB buffer
Asynchronous one client calls one server (request only)	~352,000 calls/sec @ 1 KB buffer
Synchronous service call, 5x clients calls 5x servers (request/reply)	~135,000 calls/sec @ 1 KB buffer
Bridged (two cluster nodes connected on localhost), client on one cluster node calls server on another cluster node (request/reply)	~11,000 calls/sec @ 1 KB buffer
Persistent storage, messages enqueued to solid storage	~1790 msgs/sec @ 1 KB buffer
tpcall() cached results	~975'000 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.
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 multiple transactions simultaneously on different, distributed nodes, thus eliminating single point of failure and increasing scalability.
Linear Scalability	Enduro/X provides almost linear increase in application throughput corresponding to increase in available resources. New resources can be plugged in existing cluster and its load will be balanced.
Local load balancer approach	<ul style="list-style-type: none"> 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, FreeBSD, AIX and Solaris platforms. For MacOS round-robin approach is used for dispatching requests across the same XATMI service providers.
Data Dependent Routing	Enduro/X XATMI servers may be configured in routing groups which automatically configure additional dimension for advertised services. When process makes a service call and routing expressions are defined for particular service, then system uses data in call buffer to evaluate a group a target group server to which data shall be sent. This allows at administrative level create a shards for the application.
Service call priority	On Linux on FreeBSD platforms service calls may have different priorities. Thus for the same service some critical messages can skip the queue and messages can be processed earlier.
Process level multi-threading	<p>Enduro/X XATMI server service dispatchers supports multi-threading, i.e. service routine executing in threads from thread-pool, configured by <code>mindispatchthreads/maxdispatchthreads</code> settings.</p> <p>XATMI Servers and Clients can use conventional Posix threads. These threads are initialized as Enduro/X client processes.</p>
Cluster operations load balancer approach	<p>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.</p> <p>When service is available only in cluster, then they are invoked directly no matter of the percentage settings.</p>
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 result 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

Scalability and Reliability	
	<p>caches. Cluster operations are supported - once call is cached on some cluster node, the cache data may be replicated to other nodes.</p>
Replicated service framework	<p>Administrators can dynamically replicates services across the network to maximize performance and reliability.</p>
Failover	<p>Enduro/X automatically redirects all the traffic to alive services meanwhile removing failed components from cluster visibility.</p>
Robust fault management	<p>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.</p>
Service monitoring	<p>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 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.</p>
Core fault recovery	<p>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.</p>
Real-time system patching	<p>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.</p>
System sanity	<p>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.</p> <p>In the same way, the daemon is following up the configuration and reboots any missing processes, if they have failed and exited.</p>
Logging	<p>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</p>

Scalability and Reliability	
	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 “tpsterror()” 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.
Quality Assurance	Every official Enduro/X releases passes thousands of unit tests and hundreds of integration tests. Executed on several server platforms. Where testing on each platform takes from 8h to 24h, depending on configuration.

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	<p>Enduro/X provides formal logging API named “tplog()”, “tplogdump()” 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.</p> <p>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.</p>
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.
Latent Command Framework (LCF)	Enduro/X provides administrative commands for Common logging framework. Log settings (including levels) can be changed on the fly for any Enduro/X related process. LCF also provides commands for logfile rotation. The LCF framework exposes APIs for developers for registering new administrative commands and their working callbacks via plugin interface. Thus application developers may introduce new system wide extensions for applications. For example LCF can be used to implement process notifications for requesting application specific configuration file reload, etc.

Infrastructure	
Monitoring	<ul style="list-style-type: none"> Enduro/X has Management Information Base interface – TMIB for following classes T_CLIENT, T_DOMAIN, T_MACHINE, T_QUEUE, T_SERVER, T_SERVICE, T_SVCGRP, T_BRCON (GET operation). NetXMS agent for application monitoring in NetXMS monitoring suite.
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.

Compatibility	
Oracle Tuxedo	Enduro/X is positioned as Oracle Tuxedo alternative. The compatibility grows with each of the Enduro/X major release. Core XATMI and FML APIs have high compatibility level. Migration tool “ubb2ex” allows to automatically migrate Tuxedo configuration to Enduro/X.
RedHat Blacktie	Enduro/X provides full XATMI level functionality.
Hitache OpenTP1	Provides functionality at XATMI standard level.

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 Compatibility	
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	<ol style="list-style-type: none"> 1. “,” 2. “@” 3. “/” 4. Cannot start with “.”
Maximum identifier length (service named, binary name, etc.)	30 ASCII characters.
Maximum UBF field name length	64 ASCII characters.
Maximum cluster nodes	32 out of the box, 255 theoretical limit.
Maximum IPC buffer size	Buffer size is limited to Operating System queue API limits.
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:

- √ - Supported API call.
- 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	Java	Python
Transport/IPC				
tpcall	√	√	√	√
tpacall	√	√	√	√
tpconnect	√	√	√	√
tpgetrply	√	√	√	√
tprecv	√	√	√	√
tpsend	√	√	√	√
tpreturn	√	√	√	√
tpforward	√	√	√	√
tpunadvertise	√	√	√	√
tpunsubscribe	√	√	√	√
tppost	√	√	√	√
tpdequeue	√	√	√	√
tpenqueue	√	√	√	√
tpdiscon	√	√	√	√
tpcancel	√	√	√	
tpadvertise	√	√	√	√
tpsvrdone	√	√	√	√
tpsvrinit	√	√	√	√
tpsvrthrinit	√			√
tpsvrthrdone	√			√
tpsubscribe	√	√	√	√
tpservice	√	√	√	√
tpcontinue	√	√	√	√
tpnotify	√		√	√
tpbroadcast	√		√	√
tpchkunsol	√		√	√
tpsetunsol	√		√	√
tpimport	√		√	√
tpexport	√		√	√
tpgprio	√			√
tpsprio	√			√
tpsetcallinfo	√	√		√
tpgetcallinfo	√			√
tpsbktime	√			√

XATMI Call	C/C++	Go	Java	Python
tpgblktime	√			√
Transactions				
tpbegin	√	√	√	√
tpresume	√	√	√	√
tpopen	√	√	√	√
tpclose	√	√	√	√
tpcommit	√	√	√	√
tpabort	√	√	√	√
tpgetlev	√	√	√	√
tpsuspend	√	√	√	√
tpscmt	√			√
tx_begin	√			
tx_close	√			
tx_commit	√			
tx_info	√			
tx_open	√			
tx_rollback	√			
tx_set_commit_return	√			
tx_set_transaction_control	√			
tx_set_transaction_timeout	√			
Generic Buffer Management				
tpalloc	√	√	√	(dict)
tpfree	√	GCed	GCed	(dict)
tpisautobuf	√	√	√	(dict)
tpjsontobuf	√	√	√	
tpubftojson	√	√	√	
tpypes	√	√	√	(dict)
tprealloc	√	√	√	(dict)
Extensions & hooks				
tpext_addb4pollcb	√	√		√
tpext_addperiodcb	√	√		√
tpext_addpollerfd	√	√		√
tpext_delb4pollcb	√	√		√
tpext_delperiodcb	√	√		√
tpext_delpollerfd	√	√		√
Multithreading				
tpgetctxt	√	embedded	embedded	√
tpnewctxt	√		embedded	√
tpfreectxt	√		GCed	√

XATMI Call	C/C++	Go	Java	Python
tpsetctx	√		embedded	√
tpsrvgetctxdata	√	√	√	√
tpsrvsetctxdata	√	√	√	√
Logging				
tplog	√	√	√	√
tplogclosereqfile	√	√	√	√
tplogclosethread	√	√	√	√
tplogconfig	√	√	√	√
tplogdelbufreqfile	√	√	√	√
tplogdump	√	√	√	√
tplogdumpdiff	√	√	√	√
tploggetbufreqfile	√	√	√	√
tploggetreqfile	√	√	√	√
tplogprintubf	√	√	√	√
tplogsetreqfile	√	√	√	√
tplogsetreqfile_direct	√	√	√	√
Auxiliary & integration				
ndrx_main	√			
ndrx_main_integra	√			
tpgetnodeid	√	√	√	√
tpgetsrvid	√	√	√	√
tpinit	√	√	√	√
tpsterror	√	√	Exceptions	Exceptions
tpterm	√	√	√	√

UBF API Calls

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

UBF Call	C/C++	Go	Java	Python
Field management				
Badd	√	√	√	(dict)
Badds	√	√	√	(dict)
Bchg	√	√	√	(dict)
Bchgs	√	√	√	(dict)
Bdel	√	√	√	(dict)
Bfind	√			(dict)
Bfindlast	√			(dict)

UBF Call	C/C++	Go	Java	Python
Bfindocc	√			(dict)
Bfinds	√			(dict)
Bget	√	√	√	(dict)
Bgetalloc	√			(dict)
Bgetlast	√			(dict)
Bgetsa	√			(dict)
Bgets	√			(dict)
Blen	√	√	√	(dict)
Boccur	√	√	√	(dict)
Bpres	√	√	√	(dict)
CBadd	√	√ (Badd)	√	(dict)
CBchg	√	√ (Bchg)	√	(dict)
Cbfind	√			(dict)
CBfindocc	√			(dict)
CBget	√	√ (Bget)	√	(dict)
CBgetalloc	√	√ (Bget)	√	(dict)
Field type management				
Bfldid	√	√	√	(dict)
Bfldno	√	√	√	√
Bfldtype	√	√	√	√
Bfname	√	√	√	√
Bmkfldid	√	√	√	√
Btypcvt	√	√	√	(dict)
Btype	√	√	√	(dict)
Buffer management				
Balloc	√	Tpalloc	tpalloc	(dict)
Bconcat	√	√	√	(dict)
Bcpy	√	√	√	(dict)
Bdelall	√	√	√	(dict)
Bdelete	√	√	√	(dict)
Bextread	√	√	√	√
Bfprint	√	√	√	√
Bfree	√	GCed	GCed	(dict)
Bnext	√	√	√	(dict)
Bprint	√	√	√	√
Bproj	√	√	√	(dict)
Bprojcpy	√	√	√	(dict)
Bread	√	√	√	(dict)
Brealloc	√		√	(dict)

UBF Call	C/C++	Go	Java	Python
Bsizeof	√	√	√	(dict)
Bunused	√	√	√	(dict)
Bupdate	√	√	√	(dict)
Bused	√	√	√	(dict)
Bwrite	√	√	√	(dict)
Bcmp	√		√	(dict)
Bnum	√		√	(dict)
VIEW Buffer management				
Bvftos	√			(dict)
Bvstof	√			(dict)
Bvnull	√			(dict)
Bvopt	√			(dict)
Bvselinit	√			(dict)
Bvsinit	√			(dict)
CBvget	√	√		(dict)
CBvchg	√	√		(dict)
Bvsizeof	√	√		(dict)
Bvcpy	√	√		(dict)
Bvoccur	√	√		(dict)
Bvsetoccur	√	√		(dict)
Bvnext	√	√		(dict)
Expression & scripting				
Bboolco	√	√	√	
Bboolev	√	√	√	√
Bboolpr	√	√	√	√
Bboolsetcbf	√	√	√	
Bfloatev	√	√	√	√
Btreefree	√	√	√	
Auxiliary				
Berror	√	√	Exceptions	Exceptions
Bidxused	(stub)			
Bindex	(stub)			
Brstrindex	(stub)			
Binit	√	√	√	(dict)
Bisubf	√	√	√	
Bsterror	√	√	Exception	Exceptions
Bunindex	(stub)			

Copyright © 2022, Mavimax. All rights reserved.

This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Oracle Tuxedo are trademarks or registered trademarks of Oracle Corporation. Jboss and Blacktie are trademarks or registered trademarks of Red-Hat Corporation. UNIX is a registered trademark of The Open Group. POSIX is a registered trademark of the IEEE.