NDRXCONFIG.XML(5)

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SYNOPSIS

```
<?xml version="1.0" ?>
<endurox>
    <appconfig>
        <sanity>SANITY_SECONDS</sanity>
        <checkpm>CHECKPM_STY</checkpm>
        <brrefresh>BRIDGE_REFRESH_TIME</brrefresh>
        <restart_min>MIN_RESTART_TIME</restart_min>
        <restart_step>RESTART_STEP</restart_step>
        <restart_max>MAX_RESTART_TIME</restart_max>
        <restart_to_check>NDRXD_RESTART_TO_CHECK</restart_to_check>
        <gather_pq_stats>NDRXD_GATHER_PQ_STATS</gather_pq_stats>
        <rqaddrttl>RQADDRTTL</rqaddrttl>
    </appconfig>
    <defaults>
        <min>MIN_SERVERS_DEFAULT</min>
        <max>MAX_SERVERS_DEFAULT</max>
        <autokill>AUTOKILL_DEFAULT</autokill>
        <env>ENV_OVERRIDE_DEFAULT</env>
        <start_max>MAX_STARTUP_TIME_DEFAULT</start_max>
        <pingtime>PING_EVERY_TIME_DEFAULT</pingtime>
        <ping_max>MAX_PING_TIME_DEFAULT</ping_max>
        <end_max>MAX_SERVER_SHUTDOWN_TIME_DEFAULT</end_max>
        <killtime>KILL_TIME_DEFAULT</killtime>
        <exportsvcs>EXPORT_SERVICES_DEFAULT</exportsvcs>
        <blacklistsvcs>BLACKLIST_SERVICES_DEFAULT</blacklistsvcs>
        <srvstartwait>NDRXD_SRV_START_WAIT_DEFAULT</srvstartwait>
        <srvstopwait>NDRXD_SRV_STOP_WAITT_DEFAULT</srvstopwait>
        <cctag>COMMON_CONFIG_TAG_DEFAULT</cctag>
        <protected>PROTECTED_SERVER_DEFAULT</protected>
        <reloadonchange>RELOAD_ON_CHANGE_DEFAULT</reloadonchange>
        <rssmax>RSSMAX_DEFAULT</rssmax>
        <vszmax>VSZMAX_DEFAULT</vszmax>
        <rqaddr>RQADDR_DEFAULT</rqaddr>
        <envs group="SVGRP_ENV_GROUP_NAME">
            <env name="SVGRP_ENV_VARIABLE_NAME">SVGRP_ENV_VARIABLE_VALUE</env>
            <env name="SVGRP_ENV_VARIABLE_NAME_1" unset="SVGRP_ENV_UNSET" />
            ...
            <env name="SVGRP_ENV_VARIABLE_NAME_N">SVGRP_ENV_VARIABLE_VALUE_N</env>
        </envs>
        ...
        <envs group="SVGRP_ENV_GROUP_NAME_N">
            ...
        </envs>
    </defaults>
```

```
    <servers>
        <server name="SERVER_BINARY_NAME">
            <min>MIN_SERVERS_SRV</min>
            <max>MAX_SERVERS_SRV</max>
            <autokill>AUTOKILL_SRV</autokill>
            <respawn>RESPAWN_SRV<respawn>
            <env>ENV_OVERRIDE_SRV</env>
            <start_max>MAX_STARTUP_TIME_SRV</start_max>
            <pingtime>PING_EVERY_TIME_SRV</pingtime>
            <ping_max>MAX_PING_TIME_SRV</ping_max>
            <end_max>MAX_SERVER_SHUTDOWN_TIME_SRV</end_max>
            <killtime>KILL_TIME_SRV</killtime>
            <sleep_after>SECONDS_TO_SLEEP_AFTER_SRV_START</sleep_after>
            <srvid>SERVER_ID</srvid>
            <sysopt>ATMI_SERVER_SYSTEM_OPTIONS</sysopt>
            <appopt>ATMI_SERVER_APPLICATION_OPTIONS</appopt>
            <exportsvcs>ATMI_SERVER_EXPORT_SERVICES</exportsvcs>
            <blacklistsvcs>ATMI_BLACKLIST_SERVICES</blacklistsvcs>
            <srvstartwait>NDRXD_SRV_START_WAIT</srvstartwait>
            <srvstopwait>NDRXD_SRV_STOP_WAITT</srvstopwait>
            <cctag>COMMON_CONFIG_TAG</cctag>
            <protected>PROTECTED_SERVER</protected>
            <reloadonchange>RELOAD_ON_CHANGE_SERVER</reloadonchange>
            <fullpath>ATMI_SERVER_FULL_PATH</fullpath>
            <cmdline>ATMI_SERVER_COMMAND_LINE</cmdline>
            <rssmax>ATMI_SERVER_RSSMAX</rssmax>
            <vszmax>ATMI_SERVER_VSZMAX</vszmax>
            <rqaddr>RQADDR</rqaddr>
            <envs>
                <usegroup>SVGRP_ENV_GROUP_NAME</usegroup>
                ...
                <usegroup>SVGRP_ENV_GROUP_NAME_N</usegroup>
                <env name="SVPROC_ENV_VARIABLE_NAME">SVPROC_ENV_VARIABLE_VALUE</env>
                <env name="SVPROC_ENV_VARIABLE_NAME_UNSET" unset="SVPROC_ENV_UNSET" />
                ...
                <env name="SVPROC_ENV_VARIABLE_NAME_N">SVPROC_ENV_VARIABLE_VALUE_N</env>
            </envs>
        </server>
        ...
        <server name="SERVER_BINARY_NAME_N">
            ...
        </server>
    </servers>
    <clients>
        <envs group="CLTGRP_ENV_GROUP_NAME">
            <env name="CLTGRP_ENV_VARIABLE_NAME">CLTGRP_ENV_VARIABLE_VALUE</env>
            <env name="CLTGRP_ENV_VARIABLE_NAME_UNSET" unset="CLTGRP_ENV_UNSET" />
            ...
            <env name="CLTGRP_ENV_VARIABLE_NAME_N">CLTGRP_ENV_VARIABLE_VALUE_N</env>
        </envs>
        ...
        <envs group="CLTGRP_ENV_GROUP_NAME_N">
            ...
        </envs>
        <client cmdline="CLT_COMMAND_LINE [${NDRX_CLTTAG} ${NDRX_CLTSUBSECT}]"
            log="CLT_LOG"
            stdout="CLT_STDOUT"
            stderr="CLT_STDERR"
            env="CLTGRP_ENV"
            CCTAG="CLT_CCTAG"
            wd="CLT_WD"
            klevel="CLT_KLEVEL"
```

```
            autostart="CLT_AUTOSTART"
            rssmax="CLT_RSSMAX"
            vszmax="CLT_VSZMAX"
            subsectfrom="CLT_SUBSECTFROM"
            subsectto="CLT_SUBSECTTO"
            >

            <envs>
                <usegroup>CLTGRP_ENV_GROUP_NAME</usegroup>
                ...
                <usegroup>CLTGRP_ENV_GROUP_NAME_N</usegroup>
                <env name="CLTPROC_ENV_VARIABLE_NAME">CLTPROC_ENV_VARIABLE_VALUE</env>
                <env name="CLTPROC_ENV_VARIABLE_NAME_UNSET" unset="CLTPROC_ENV_UNSET" />
                ...
                <env name="CLPROC_ENV_VARIABLE_NAME_N">CLTPROC_ENV_VARIABLE_VALUE_N</env>
            </envs>

            <exec tag="CLT_TAG_EXEC"
                subsect="CLT_SUBSECT_EXEC"
                log="CLT_LOG_EXEC"
                stdout="CLT_STDOUT_EXEC"
                stderr="CLT_STDERR_EXEC"
                env="CLTGRP_ENV_EXEC"
                cctag="CLT_CCTAG_EXEC"
                wd="CLT_WD_EXEC"
                autostart="CLT_AUTOSTART_EXEC"
                klevel="CLT_KLEVEL_EXEC"
                rssmax="CLT_RSSMAX_EXEC"
                vszmax="CLT_VSZMAX_EXEC"
                subsectfrom="CLT_SUBSECTFROM_EXEC"
                subsectto="CLT_SUBSECTTO_EXEC"
                />
            <exec tag="CLT_TAG_EXEC2"
                subsect="CLT_SUBSECT2_EXEC2"
                .../>
        </client>
        <client cmdline="BINARY2" ...>
            <exec tag="CLT_EXE_TAG2" .../>
        </client>
    <clients>
</endurox>
```

Chapter 2

DESCRIPTION

ndrxconfig.xml holds the application domain configuration. It describes the ATMI servers which needs to be started. Counts of the, how much to start. Also it describes sanity times i.e. period after which system sanity checks should be made. Also it describes time frames in which ATMI server should start or stop. Internal server ping can be configured here too.

Chapter 3

PARAMETERS

SANITY_SECONDS
Number of seconds after which perform system sanity checks. This number should divide by environment variable value *NDRX_CMDWAIT*. As this actually is time by which *ndrxd* sleeps periodically.

CHECKPM_STY
This is number of sanity cycles into which check dead processes from the process model. This makes the actual checking of th PID existence system. Thus if ndrxd is started in learning mode and will not receive signals of the dead servers, then by setting it will discover exited processes.

BRIDGE_REFRESH_TIME
Number of sanity units in which *tpbridge* refresh should be send to other node. If for example SANITY_SECONDS is set to 10, and BRIDGE_REFRESH_TIME is set to 2 then period between bridge refreshes will be 10*2 = 20 seconds. Default value is **0** - do not send full updates.

MIN_RESTART_TIME
Number of sanity units in which died server will be tried to start back. This is minimal time, means that this time is applied in case if server was running and died. If it is consecutive try, then RESTART_STEP is applied on this timer.

RESTART_STEP
Number to sanity units to apply on *MIN_RESTART_TIME* in case of consecutive server death. Meaning that next try of restart will tried later that previous by this number of sanity units.

MAX_RESTART_TIME
Max number of sanity units after which server will tried to restart. After each consecutive ATMI server death, next reboot is tried by *MIN_RESTART_TIME+RESTART_STEP**try_count. If this goes over the *MAX_RESTART_TIME* then *MAX_RESTART_TIME* is used instead.

NDRXD_RESTART_TO_CHECK
Number of **seconds** for *ndrxd* to wait after daemon started in recovery mode. Within this time no sanity checks are perfomed, but instead "learning" mode is used. During this mode, *ndrxd* asks each ATMI server for it's configuration. If in this time ATMI server does not responds, then ATMI server is subject of sanity checks.

NDRXD_GATHER_PQ_STATS
Settings for **pq** *xadmin* command. if set to *Y*, ndrxd will automatically collect stats for service queues. In future this might be used for automatic service starting and stopping.

RQADDRTTL
Used only when operating System V queues mode. Due to common queue for multiple services / basically all service queues are shared request addresses, the only zapping approach when there are no servers on queues, is to check that in service shared memory there are no linked request address queues, and at time perform unlink of the request address queue. But here we have a problem. The XATMI server might just started up, opened the RQADDR queue, but did not yet managed to install record in service shared memory. Thus *ndrxd* will unlink the RQADDR. To avoid this issue, with

TTL slight delay is introduced, after which queue is unlinked. Basically when queue is open it's change time is updated. And if current time minus change time is less than **RQADDRTTL**, then queue is not unlinked (in this time server will be able to add record to service shared memory). Also with this comes a fact that there must be no server processes with out any service. For those request address queue will be unlinked. The value is in seconds. Checks are performed with **SANITY_SECONDS** intervals. Default value is **10** seconds.

MIN_SERVERS_DEFAULT
Default minimum number of copies of the server which needs to be started automatically. This can be overridden by *MIN_SERVERS_SRV* per server.

MAX_SERVERS_DEFAULT
Max number of ATMI server copies per ATMI server entry. The difference between MIN and MAX servers means the number of standby servers configured. They can be started by hand with out system re-configuration. But they are not booted automatically at system startup. You will have to start them with $ xadmin start -s <server_name> or by $ xadmin start -i <server_id>. This can be overridden by *MAX_SERVERS_SRV*.

AUTOKILL_DEFAULT
Should server be automatically killed (by sequence signal sequence -2, -15, -9) in case if server have been starting up too long, or does not respond to pings too long, or it is performing shutdown too long. This can be overridden by *AUTOKILL_SRV* on per server basis.

ENV_OVERRIDE_DEFAULT
Full path to file containing environment variable overrides. see *ex_envover(5)* for more details. This can be overridden by per server basis by ENV_OVERRIDE_SRV. Both are optional settings.

MAX_STARTUP_TIME_DEFAULT
Max time (in sanity units) in which server should start up, i.e. send init info to *ndrxd*. If during this time server have not initialized, it is being restarted. This can be overridden by *MAX_STARTUP_TIME_SRV*.

PING_EVERY_TIME_DEFAULT
Number of sanity units in which perform periodical server pings. This can be overridden by *PING_EVERY_TIME_SRV*. Zero value disables ping.

MAX_PING_TIME_DEFAULT
Number of sanity units, time in which server **must** respond to ping requests. If there is no response from server within this time, then restart sequence is initiated. This can be overridden by *MAX_PING_TIME_SRV*.

MAX_SERVER_SHUTDOWN_TIME_DEFAULT
Maximum time in which shutdown of server must complete in sanity units. If in given time server is not shutdown, then forced shutdown sequence is started until server exits. This can be overridden by *MAX_SERVER_SHUTDOWN_TIME_SRV* on per server basis.

EXPORT_SERVICES_DEFAULT
Comma separated list of services to be applied to all binaries which means the list of services to be exported by **tpbridge** server to other cluster node. This can be overridden by *ATMI_SERVER_EXPORT_SERVICES*.

BLACKLIST_SERVICES_DEFAULT
Comma separated list of services to be applied to all server binaries which means the list of services that must not be exported by **tpbridge** server to other cluster node. *ATMI_SERVER_BLACKLIST_SERVICES* is first priority over the *EXPORT_SERVICES_DEFAULT* if service appears in both lists. *BLACKLIST_SERVICES_DEFAULT* can be overridden by *ATMI_SERVER_BLACKLIST_SERVICES*.

NDRXD_SRV_START_WAIT_DEFAULT
Number of seconds to wait for servers to boot. If not started in given time, then continue with next server. This can be overridden by *NDRXD_SRV_START_WAIT*. Default value for this is 30 seconds.

NDRXD_SRV_STOP_WAIT_DEFAULT
Number of seconds to wait for server to shutdown. If not started in given time, then continue with next server. This can be overridden by *NDRXD_SRV_STOP_WAIT_DEFAULT*. Default value for this is 30 seconds.

KILL_TIME_DEFAULT
Time in sanity units after which to progress from first signal -2 to next signal -15. And after -15 this time means when next -9 signal will be sent. This is used if forced restart of forced shutdown was initiated by *ndrxd*. This can be overridden by *KILL_TIME_SRV*.

COMMON_CONFIG_TAG_DEFAULT
Common configuration tag. Loaded into *NDRX_CCTAG* environment variable before process is spawned. This can be overridden by *COMMON_CONFIG_TAG*.

PROTECTED_SERVER_DEFAULT
Protected server is one that does not shutdown with *xadmin stop* unless you pass the *xadmin stop -c* parameter (complete shutdown). Still you can run the *sreload* and stop it by *xadmin stop -i <srvid>* or by *xadmin stop -s <servernm>*. The *xadmin restart* won't work on these because *-c* is not supposed to be used by restart. The idea behind this, is to avoid accidental stop of the critical servers, like bridge or something else which is involved into *ndrxd* daemon management it self. This can be overridden by *PROTECTED_SERVER*.

RELOAD_ON_CHANGE_DEFAULT
If set to **Y** or **y** the **ndrxd** daemon will scan the every binaries time stamp, and if it detects that time stamp is changed **ndrxd** will reload (stop/start) the XATMI servers one by one. The scanning will occur at every sanity cycle. This is recommended to be used **only** for development purposes. And must not be used on production servers! This can be overridden by *RELOAD_ON_CHANGE_SERVER* on per server basis.

RSSMAX_DEFAULT
Maximum Resident Set Size memory size after which **ndrxd(8)** process will issue server reload (sr) command (i.e. gracefully shutdown and start back) to server process if particular server process resident memory goes over this defined value. The value can be override by **ATMI_SERVER_RSSMAX** for particular server instance. This parameter is useful to be used to protect local machine against defective/binaries with memory leaks. The parameter value is expressed in bytes. Configuration file also accepts suffixes such as "T" or "t" for terrabytes, "G" or "g" for gigabytes, "M" or "m" for megabytes and "K" or "k" for kilobytes. For example "10M" would limit resident memory to 10 megabytes. The default value is **-1**, which means that functionality is not used.

VSZMAX_DEFAULT
Maximum Virtual Set Size memory size (the number bytes program have asked to OS kernel for memory, but does it does **not** mean it is physically used or initialized) after which **ndrxd(8)** process will issue server reload (sr) command (i.e. gracefully shutdown and start back) to server process if particular server's process virtual memory goes over this defined value. The value can be override by **ATMI_SERVER_VSZMAX** for particular server instance. This parameter is useful to be used to protect local machine defective/binaries with memory leaks. The parameter value is expressed in bytes. Configuration file also accepts suffixes such as "T" or "t" for terrabytes, "G" or "g" for gigabytes, "M" or "m" for megabytes and "K" or "k" for kilobytes. For example "10M" would limit resident memory to 10 megabytes. The default value is **-1**, which means that functionality is not used.

RQADDR_DEFAULT
Request address (common service queue) used in System V mode. For other modes each service have it's own queue, but due to limitations of the System V queues, for each XATMI server process have it's own queue (built as process /exe name and service id) or processes can share the queue by having this request address, thus getting a one queue multiple servers mechanism for message dispatching. Also all servers attached on the same request address must advertise all the services from all servers attached on the same request address. If some server will miss a service, it might receive request for particular service, the error will be logged and message will be dropped, thus caller will get a timeout. If different request addresses are serving the same service, then request will be load balanced in round-robin mode. This can be overridden by *RQADDR* on per server basis. Request address cannot start with @ symbol. The max length of the request address is **30** chars.

SECONDS_TO_SLEEP_AFTER_SRV_START
Number of seconds to wait for next item to start after the server is launched. This is useful in cases when for example we start bridge server, let it for some seconds to connect to other node, then continue with other service startup.

SERVER_BINARY_NAME
ATMI server executable's name. The executable must be in $PATH. This name cannot contain special symbols like path separator / and it cannot contains commas ,! Commas are used as internal queue separator combined with binary names.

RESPAWN_SRV

Do the automatic process re-spawning if process is died for some reason. The default value is **Y**, meaning that processes are automatically recovered. If set to *N* or *n*, then sanity checks will not automatically re-boot the process.

SERVER_ID

Server ID. It is internal ID for server instance. For each separate ATMI server the ID must be unique. Also special care should be take when MAX_SERVERS_SRV is greater than 1. In this case up till MAX servers, internally *SERVER_ID* is incremented. Thus for example if SERVER_ID is 200, and MAX_SERVERS_SRV is 5, then following server IDs will be reserved: 200, 201, 202, 203, 204. The maximum server id is set in $NDRX_SRVMAX environment variable. Minimal server id is 1.

ATMI_SERVER_SYSTEM_OPTIONS

Command line system options passed to ATMI server. Following parameters are used by Enduro/X ATMI servers: *-N*, boolean type. If present, then no services will be advertised by server. In this case will be advertised only services specified by *-s* flag. For example if server advertises SERVICE1, SERVICE2, SERVICE3, but *-N* was specified, and *-sSERVICE3* is specified, then only service SERVICE3 will be advertised. The *-s* argument also can contain aliases for services, for example *-sOTHERSVC:SERVICE2*, then new service OTHERSVC will be advertised which basically is the same SERVICE2 (same function used). *-s* and *-N* can be mixed. *-s* can appear multiple times in system options. With one *-s* multiple services can be aliased to single existing service. The format is: *-s<NEWSVC1>/<NEWSVC2>/../<NEWSVCN>:<EXISTINGSVC>*. The *,* can be used as separator too, but for certain platforms it does not work, thus */* is recommended. When using full advertise (all service) some of them can be masked by *-n* flag. For example *-sSERVICE4* will advertise all, but *SERVICE4*. Flag *-n* can be repeated multiple times. Server binaries output is controlled via *-e LOG_FILE*, which means that stdout & stderr of server is dumped to LOG_FILE. There are few internal params: param *-k* is just a random key for shell scripts. Another internal param is Server ID which is automatically passed to binary via *-i SERVER_ID*. Enduro/X supports automatic buffer conversion for ATMI servers. Currently supported modes are *JSON2UBF*, *UBF2JSON*, these modes are activated by -x paramter in system options. These modes are passed for server functions being advertised. For example if we have service **functions** (not services) UBF1FUNC, UBF2FUNC and JSONFUNC and we want to ensure that these receive converted messages even if caller to UBF service sends JSON and vice versa, then following options might be set to command line: *-xUBF1FUNC,UBF2FUNC:JSON2UBF -xJSONFUNC:UBF2JSON*. When Enduro/X is build with integration lib (libatmisrvinteg) user have ability to map the functions with -S <service_name1>/<service_name2>:<function name> passed to buildserver at build time or registered in tmdsptchtbl_t array passed to _tmstartserver(). The -S are processed and service array is prepared, afterwards the -N and -s<service_name1> flag works as usual, with -N disable all service advertise and with -s some single service may be enabled.

ATMI_SERVER_APPLICATION_OPTIONS

Application specific command line options. This follows content after sys options as: *system options — app options*.

ATMI_SERVER_EXPORT_SERVICES

Enduro/X server specific list of services to be exported. This list is only for **tpbridge** servers.

ATMI_SERVER_BLACKLIST_SERVICES

Enduro/X server specific list of services that must not be exported. This list is only for **tpbridge** servers. Blacklist have higher priority over the Export list.

ATMI_SERVER_FULL_PATH

This is full path of the XATMI server binary. At the process startup this overrides the server binary name at **SERVER_BINARY_N**
ATMI_SERVER_FULL_PATH is used only for process startup. This is intended for testing, if server wrapper scripts needs to be started. But as the **ndrxd** will do the sanity checks against the process names, for time of the testing this needs to be disabled. Thus to do the testings with full path enabled, please increase the *checkpm* sanity unit time. As at the moment of process model checks, the ndrxd will find out that wrapped binary name does not contain the **SERVER_BINARY_NAME**, thus will reboot the process.

ATMI_SERVER_COMMAND_LINE

This is alternative command line build by user. From this command line the real process name is extracted as first executable (basename). When building custom command line, the env substitution is available at the stage with following processes based envs (not counting the globals): **NDRX_SVSRVID** - Enduro/X server id, **NDRX_SVPROCNAME** - server process name (defined in XML config as **SERVER_BINARY_NAME** variable, **NDRX_SVCLOPT** - standard command line options used by Enduro/X. These options are used at stage with ATMI server library gets initialized, it will use in case **ndrx_main()** receives less than expected standard argument count. Basically this command line tag is

suitable for interpreted languages, like Java, where interpreter needs to be started as stand alone binary, and the Enduro/X is initialized as a library within stand alone process.

SVGRP_ENV_GROUP_NAME
Environment variable group name for the servers section. Identifier max length is 30 chars. Same group can be used for different server processes. One server may import multiple groups. For client processes groups are defined **CLTGRP_ENV_GROUP_NAME** name at <*clients*> section. At process level groups can be imported by using tag <*usegroup*> and specifying the group name. At that moment all variables defined in group are import for process.

SVGRP_ENV_VARIABLE_NAME
This server server's group environment variable name, that shall be set for process which uses this group. For client processes **CLTGRP_ENV_VARIABLE_NAME** set the variable name at group definition. Individual environment variables can be set at process level. For server processes that is set by **SVPROC_ENV_VARIABLE_NAME** and for client processes by **CLTPROC_ENV_VARIABLE_NAME**.

SVGRP_ENV_VARIABLE_VALUE
This is environment variable value to be set. For client process groups this is defined by **CLTGRP_ENV_VARIABLE_VALUE**. For individual processes value is defined by **SVPROC_ENV_VARIABLE_NAME** and **CLTPROC_ENV_VARIABLE_NAME** accordingly. The value is interpreted by variable substitution algorithm (see bellow). The value is interpreted at time when process is spawned (not defined), meaning that it have access to full process variables at startup moment.

SVGRP_ENV_UNSET
If set to *y* or *Y* then environment's environment variable is unset (removed) from environment. This can be used if some specific variable for process is not needed. At client environment group level this can be set by **CLTGRP_ENV_UNSET**, at process levels this can be set by **SVPROC_ENV_UNSET** and **CLTPROC_ENV_UNSET** accordingly. If any value is present for this variable, it is ignored, as the main action of this tag is unset the value and only what matters here is the variable name.

CLT_COMMAND_LINE
Executable name and arguments for client program. Command line basically is a format for subsection substitution. Other env variables available here too.

CLT_LOG
Logfile to which stdout and stderr is logged. Can be overridden by *CLT_LOG_EXEC* for each individual process. Optional attribute.

CLT_STDOUT
File where to log stdout. Can be overridden by *CLT_STDOUT_EXEC* for each individual process. Optional attribute.

CLT_STDERR
File where to log stderr. Can be overridden by *CLT_STDERR_EXEC* for each individual process. Optional attribute.

CLTGRP_ENV
Environment override file. See **ex_envover(5)** for syntax. Can be overridden by *CLTGRP_ENV_EXEC* for each individual process. Optional attribute.

CLT_CCTAG
ATMI Client lib Common-Config tag. Can be overridden by *CLT_CCTAG_EXEC* for each individual process. Optional attribute.

CLT_WD
Working directory for the process. Can be overridden by *CLT_WD_EXEC*.

CLT_AUTOSTART
Should process be started automatically? *Y* or *y* means boot at start. Can be overridden by *CLT_AUTOSTART_EXEC* for each individual process. Optional attribute. Default *n*.

CLT_TAG_EXEC
Tagname to be set for process.

CLT_SUBSECT_EXEC
Subsection to be set for process. - used as default.

CLT_RSSMAX

Maximum Resident Set Size memory size after which **cpmsrv(8)** process will gracefully kill the client process by signals -2, -15, -9 if particular client process resident memory goes over this defined value. After killing, the cpmsrv at first check interval will detect that client is dead, and at next check interval it will be respawned. The value can be override by **CLT_RSSMAX_EXEC**. This parameter is useful to be used to protect local machine against defective/binaries with memory leaks. The parameter value is expressed in bytes. Configuration file also accepts suffixes such as "T" or "t" for terrabytes, "G" or "g" for gigabytes, "M" or "m" for megabytes and "K" or "k" for kilobytes. For example "10M" would limit resident memory to 10 megabytes. The default value is **-1**, which means that functionality is not used.

CLT_VSZMAX

Maximum Virtual Set Size memory size (the number bytes program have asked to OS kernel for memory, but does it does **not** mean it is physically used or initialized) after which **cpmsrv(8)** process will gracefully kill the client process by signals -2, -15, -9 if particular client process virtual memory goes over this defined value. After killing, the cpmsrv at first check interval will detect that client is dead, and at next check interval it will be respawned. The value can be override by **CLT_RSSMAX_EXEC**. This parameter is useful to be used to protect local machine against defective/binaries with memory leaks. The parameter value is expressed in bytes. Configuration file also accepts suffixes such as "T" or "t" for terrabytes, "G" or "g" for gigabytes, "M" or "m" for megabytes and "K" or "k" for kilobytes. For example "10M" would limit resident memory to 10 megabytes. The default value is **-1**, which means that functionality is not used.

CLT_SUBSECTFROM

If set, then used to auto-generate number subsections for particular client definition. In this case **CLT_SUBSECT_EXEC** parameter value is ignored. The loop must start with non negative number and must not be greater than integer (2147483647) and less or equal to **CLT_SUBSECTTO**. Default is **undefined** and not used. Can be overriden by **CLT_SUBSECTFROM_EXEC** Must be used together with **CLT_SUBSECTTO**.

CLT_SUBSECTTO

If set, then used to auto-generate number subsections for particular client definition. In this case **CLT_SUBSECT_EXEC** parameter value is ignored. The loop must end with non negative number and must not be greater than integer (2147483647). Default is **undefined** and not used. Can be overriden by **CLT_SUBSECTFROM_EXEC**. Must be used together with **CLT_SUBSECTFROM**.

CLT_KLEVEL

Kill level of the client. **0** - do not kill child processes recursively of the client, **1** - do kill child processes only when performing SIGKILL (-9), **2** - do kill on SIGTERM and SIGINT child processes. The default is **0**.

Chapter 4

VARIABLE SUBSTITUTION

Several parameters in the ndrxconfig.xml file are processed via substitution engine. Engine processes puts the environment variables or special functions variables and $[PARAMETER] for functions. The value can be escaped with

Functions are processed in case if statement in brackets contains equal sign =. As the sign is not allowed for environment variables, Enduro/X uses it to distinguish between env variable and function.

Following **FUNC** (functions) are defined:

dec
: Decrypt base64 string in **PARAMETER** and replace the placeholder with the value. To get encrypted value, it is possible to use **exencrypt(8)** tool.

Chapter 5

EXAMPLE

Sample configuration:

```
<?xml version="1.0" ?>
<endurox>
    <appconfig>
        <sanity>10</sanity>
        <brrefresh>6</brrefresh>
        <restart_min>1</restart_min>
        <restart_step>1</restart_step>
        <restart_max>5</restart_max>
        <restart_to_check>20</restart_to_check>
    </appconfig>
    <defaults>
        <min>1</min>
        <max>2</max>
        <autokill>1</autokill>
        <start_max>2</start_max>
        <pingtime>1</pingtime>
        <ping_max>4</ping_max>
        <end_max>3</end_max>
        <killtime>1</killtime>
        <envs group="JAVAENV">
            <env name="_JAVA_OPTIONS">-Xmx1g</env>
        </envs>
    </defaults>
    <servers>
        <server name="tpevsrv">
            <srvid>14</srvid>
            <min>1</min>
            <max>1</max>
            <cctag>RM1</cctag>
            <env>${NDRX_HOME}/tpevsrv_env</env>
            <sysopt>-e /tmp/TPEVSRV -r</sysopt>
        </server>
        <server name="tpbridge">
            <max>1</max>
            <srvid>100</srvid>
            <sysopt>-e /tmp/BRIDGE -r</sysopt>
            <appopt>-n2 -r -i 0.0.0.0 -p 4433 -tA</appopt>
        </server>
        <server name="jserver2">
            <max>1</max>
            <srvid>200</srvid>
            <sysopt>-e /tmp/BRIDGE -r</sysopt>
```

```
            <envs>
                <usegroup>JAVAENV</usegroup>
                <env name="CLASSPATH">${NDRX_APPHOME}/libs/somelib.jar</env>
                <env name="CLASSPATH">${CLASSPATH}:${NDRX_APPHOME}/libs/${NDRX_SVPROCNAME}. ↩
                    jar</env>
            </envs>
            <cmdline>java</cmdline>
        </server>
        <server name="cpmsrv">
            <cctag>RM2</cctag>
            <srvid>9999</srvid>
            <sysopt>-e /tmp/cpmsrv.log -r -- -k3 -i1</sysopt>
        </server>
    </servers>
    <clients>
        <client cmdline="testbinary -t ${NDRX_CLTTAG} -s ${NDRX_CLTSUBSECT}" autostart="Y"  ↩
            cctag="RM4">
            <exec tag="TAG1" subsect="SUBSECTION1" log="${APP_LOG}/testbin1-1.log" cctag=" ↩
                RM5"/>
            <exec tag="TAG2" subsect="SUBSECTION2" log="${APP_LOG}/testbin1-2.log"/>
        </client>
        <client cmdline="testenv.sh" env="environment.override1" log="env1.log">
            <exec tag="TESTENV" autostart="Y"/>
        </client>
    </clients>
</endurox>
```

Chapter 6

BUGS

Report bugs to support@mavimax.com

Chapter 7

SEE ALSO

xadmin(8), **ndrxd(8)**, **ndrxconfig.xml(5)**, **ndrxdebug.conf(5)**, **ex_envover(5)**, **exencrytp(8)**

Chapter 8

COPYING