ngx_resolver.h Documentation:

Macros Defined:

```
#define NGX_RESOLVE_A              1
#define NGX_RESOLVE_CNAME      5
#define NGX_RESOLVE_PTR           12
#define NGX_RESOLVE_MX            15
#define NGX_RESOLVE_TXT           16
#define NGX_RESOLVE_AAAA        28
#define NGX_RESOLVE_DNAME     39
#define NGX_RESOLVE_FORMERR   1
#define NGX_RESOLVE_SERVFAIL    2
#define NGX_RESOLVE_NXDOMAIN  3
#define NGX_RESOLVE_NOTIMP     4
#define NGX_RESOLVE_REFUSED    5
#define NGX_RESOLVE_TIMEDOUT  NGX_ETIMEDOUT
#define NGX_NO_RESOLVER       (void *) -1
#define NGX_RESOLVER_MAX_RECURSION    50
```

Data Structures Defined:

```
typedef struct {
    ngx_connection_t     *connection;
    struct sockaddr        *sockaddr;
    socklen_t                  socklen;
    ngx_str_t                  server;
    ngx_log_t                  log;
} ngx_udp_connection_t;

struct ngx_resolver_ctx_s {
    ngx_resolver_ctx_t       *next;
    ngx_resolver_t             *resolver;
    ngx_udp_connection_t     *udp_connection;
    ngx_int_t                 ident;
    ngx_int_t                 state;
    ngx_str_t                 name;
    ngx_uint_t                naddrs;
    ngx_addr_t                *addrs;
    ngx_addr_t                addr;
    struct sockaddr_in        sin;
    ngx_resolver_handler_pt   handler;
    void                     *data;
    ngx_msec_t           timeout;
    ngx_uint_t            quick;
    ngx_uint_t            recursion;
    ngx_event_t          *event;
};

typedef struct {
    ngx_rbtree_node_t         node;
    ngx_queue_t               queue;
    u_char                   *name;

    #if (NGX_HAVE_INET6)
        struct in6_addr           addr6;
    #endif

    u_short                   nlen;
    u_short                   qlen;
    u_char                   *query;

    #if (NGX_HAVE_INET6)
```

```
        u_char                *query6;
    #endif

    union {
        in_addr_t             addr;
        in_addr_t            *addrs;
        u_char               *cname;
    } u;

    u_char                    code;
    u_short                   naddrs;
    u_short                   cnlen;

   #if (NGX_HAVE_INET6)
       union {
           struct in6_addr       addr6;
           struct in6_addr      *addrs6;
       } u6;
    u_short                   naddrs6;
    #endif

    time_t                    expire;
    time_t                    valid;
    uint32_t                  ttl;
    ngx_resolver_ctx_t       *waiting;
} ngx_resolver_node_t;
…………………………………………………

typedef struct {
    ngx_event_t          *event;
    void                 *dummy;
    ngx_log_t            *log;
    ngx_int_t             ident;
    ngx_array_t           udp_connections;
    ngx_uint_t            last_connection;
    ngx_rbtree_t          name_rbtree;
    ngx_rbtree_node_t     name_sentinel;
    ngx_rbtree_t          addr_rbtree;
    ngx_rbtree_node_t     addr_sentinel;
    ngx_queue_t           name_resend_queue;
    ngx_queue_t           addr_resend_queue;
    ngx_queue_t           name_expire_queue;
    ngx_queue_t           addr_expire_queue;

   #if (NGX_HAVE_INET6)
       ngx_uint_t            ipv6;
       ngx_rbtree_t          addr6_rbtree;
       ngx_rbtree_node_t     addr6_sentinel;
       ngx_queue_t           addr6_resend_queue;
       ngx_queue_t           addr6_expire_queue;
   #endif

    time_t                resend_timeout;
    time_t                expire;
    time_t                valid;
    ngx_uint_t            log_level;
} ngx_resolver_t;
```

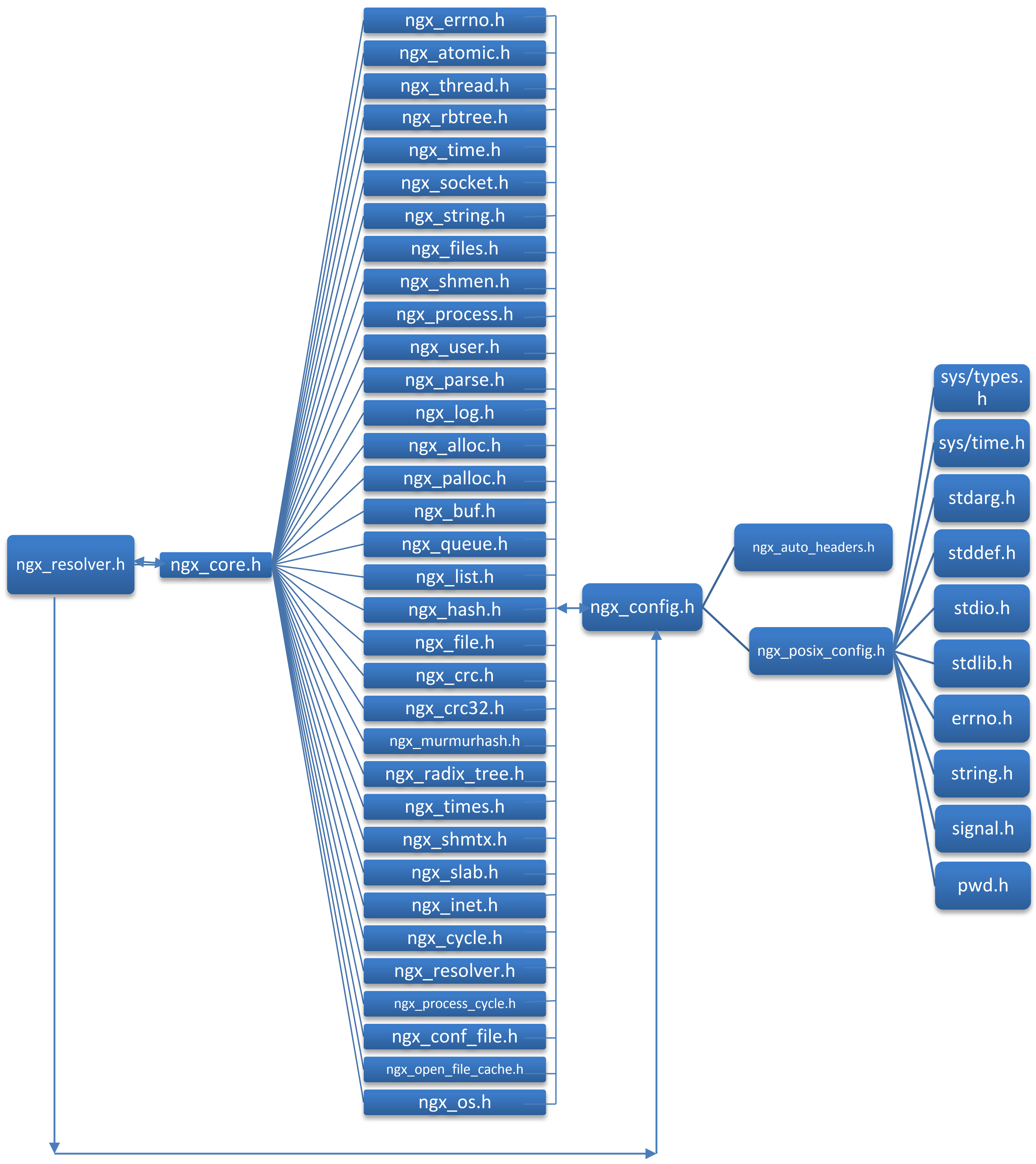
Type Definitions:

```
typedef struct ngx_resolver_ctx_s  ngx_resolver_ctx_t;
typedef void (*ngx_resolver_handler_pt)(ngx_resolver_ctx_t *ctx);
```

Functions Declared:

```
ngx_resolver_t *ngx_resolver_create( ngx_conf_t *cf,
                                     ngx_str_t *names,
                                     ngx_uint_t n
                                   );

ngx_resolver_ctx_t *ngx_resolve_start( ngx_resolver_t *r,
                                       ngx_resolver_ctx_t *temp
                                     );

ngx_int_t ngx_resolve_name( ngx_resolver_ctx_t *ctx);

void ngx_resolve_name_done(ngx_resolver_ctx_t *ctx);

ngx_int_t ngx_resolve_addr(ngx_resolver_ctx_t *ctx);

void ngx_resolve_addr_done(ngx_resolver_ctx_t *ctx);

char *ngx_resolver_strerror(ngx_int_t err);
```

Include Dependency Graph :


ngx_resolver.h
ngx_core.h
ngx_errno.h
ngx_atomic.h
ngx_thread.h
ngx_rbtree.h
ngx_time.h
ngx_socket.h
ngx_string.h
ngx_files.h
ngx_shmen.h
ngx_process.h
ngx_user.h
ngx_parse.h
ngx_log.h
ngx_alloc.h
ngx_palloc.h
ngx_buf.h
ngx_queue.h
ngx_list.h
ngx_hash.h
ngx_file.h
ngx_crc.h
ngx_crc32.h
ngx_murmurhash.h
ngx_radix_tree.h
ngx_times.h
ngx_shmtx.h
ngx_slab.h
ngx_inet.h
ngx_cycle.h
ngx_resolver.h
ngx_process_cycle.h
ngx_conf_file.h
ngx_open_file_cache.h
ngx_os.h
ngx_config.h
ngx_auto_headers.h
ngx_posix_config.h
sys/types.h
sys/time.h
stdarg.h
stddef.h
stdio.h
stdlib.h
errno.h
string.h
signal.h
pwd.h