**GOODREADS API METHODS**

# **OUATH REQUIRED:**

# **Get id of user who authorized OAuth (xml)** **URL:** [https://www.goodreads.com/api/auth\_user](https://www.goodreads.com/api/index) **HTTP method:** GET

# **Follow an author**

**URL:** [https://www.goodreads.com/author\_followings?id=AUTHOR\_ID&format=xml](https://www.goodreads.com/api/index) **HTTP method:** POST

# **Unfollow an author**

**URL:** [https://www.goodreads.com/author\_followings/AUTHOR\_FOLLOWING\_ID?format=xml](https://www.goodreads.com/api/index)

**HTTP method:** DELETE

**Show author following information (xml)**  
**URL:** [https://www.goodreads.com/author\_followings/AUTHOR\_FOLLOWING\_ID?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET

# **Compare books with another member**

Get an xml response with stats comparing your books to another member's [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/user/compare/1.xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET  
**Parameters:**

* id: Goodreads user\_id you want to compare books to

# **Get a user's followers (xml)**

Get an xml response with the given user's followers [using OAuth](https://www.goodreads.com/api/documentation#oauth).  
**URL:** [https://www.goodreads.com/user/USER\_ID/followers.xml](https://www.goodreads.com/user/18207016/followers.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/user/18207016/followers.xml?key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:** key: [Developer key](https://www.goodreads.com/api/keys) (required).page: 1-N (optional, default 1)

# **Get people a user is following (xml)**

Get an xml response with people the given user is following [using OAuth](https://www.goodreads.com/api/documentation#oauth).  
**URL:** [https://www.goodreads.com/user/USER\_ID/following.xml](https://www.goodreads.com/user/18207016/following.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/user/18207016/following.xml?key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:** key: [Developer key](https://www.goodreads.com/api/keys) (required), page: 1-N (optional, default 1)

# **Get a user's friends (xml)** **URL:** [https://www.goodreads.com/friend/user.xml](https://www.goodreads.com/friend/user/18207016?format=xml)    [(sample url)](https://www.goodreads.com/friend/user/18207016?format=xml) **HTTP method:** GET **Parameters:** id: Goodreads user\_id (required), page: 1-N (optional, default 1), sort: first\_name|date\_added|last\_online (optional)

# **Update user status** **URL:** [https://www.goodreads.com/user\_status.xml](https://www.goodreads.com/api/index) **HTTP method:** POST **Parameters:**

* user\_status[book\_id]: id of the book being reviewed (optional)
* user\_status[page]: page of the book (optional)
* user\_status[percent]: percent complete (use instead of page if appropriate)
* user\_status[body]: status update (required, unless page or percent is present, then it is optional)

# **Delete user status** **URL:** [https://www.goodreads.com/user\_status/destroy/USER\_STATUS\_ID?format=xml](https://www.goodreads.com/api/index) **HTTP method:** POST

# **Add a book to a shelf**

Add a book to a shelf [using OAuth](https://www.goodreads.com/api/documentation#oauth). This method can also be used to remove from shelf. You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/shelf/add\_to\_shelf.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* name: Name of the shelf (see: [shelves.list](https://www.goodreads.com/api/index" \l "shelves.list))
* book\_id: id of the book to add to the shelf
* a: Leave this blank unless you're removing from a shelf. If removing, set this to 'remove'. (optional)

# **Add books to many shelves**

Add a list of books to many current user's shelves [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/shelf/add\_books\_to\_shelves.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* bookids: comma-separated list of book ids
* shelves: comma-separated list of shelf names

# **Get a user's shelves**

Lists shelves for a user  
**URL:** [https://www.goodreads.com/shelf/list.xml](https://www.goodreads.com/shelf/list.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/shelf/list.xml?key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET  
**Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* user\_id: Goodreads user id (required)
* page: 1-N (default 1)

# **OUATH NOT REQUIRED:**

# **Paginate an author's books (xml)** **URL:** [https://www.goodreads.com/author/list.xml](https://www.goodreads.com/author/list/18541?format=xml&key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/author/list/18541?format=xml&key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET **Parameters:** key: [Developer key](https://www.goodreads.com/api/keys), id: Goodreads Author id (required), page: 1-N (default 1)

# **Get info about an author by id (xml)**

**URL:** [https://www.goodreads.com/author/show.xml](https://www.goodreads.com/author/show/18541?format=xml&key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:** key, id: Goodreads Author id.

# **Get Goodreads book IDs given ISBNs** **URL:** [https://www.goodreads.com/book/isbn\_to\_id](https://www.goodreads.com/book/isbn_to_id/0441172717,0739467352?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/book/isbn_to_id/0441172717,0739467352?key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET **Parameters:** key, isbn: ISBNs of books to look up.

# **Get Goodreads work IDs given Goodreads book IDs**

Get Goodreads work IDs given one or more Goodreads book IDs. Response contains work IDs without any markup.  
**URL:** [https://www.goodreads.com/book/id\_to\_work\_id](https://www.goodreads.com/book/id_to_work_id/1842,1867?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/book/id_to_work_id/1842,1867?key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:** key: [Developer key](https://www.goodreads.com/api/keys) (required), id: Book IDs of books to look up.

# **Get review statistics given a list of ISBNs**

Get review statistics for books given a list of ISBNs. ISBNs can be specified as an array (e.g. isbns[]=0441172717&isbns[]=0141439602) or a single, comma-separated string (e.g. isbns=0441172717,0141439602). You can mix ISBN10s and ISBN13s, but you'll receive a 422 error if you don't specify any, and you'll receive a 404 if none are found.  
**URL:** [https://www.goodreads.com/book/review\_counts.json](https://www.goodreads.com/book/review_counts.json?isbns=0441172717%2C0141439602&key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/book/review_counts.json?isbns=0441172717%2C0141439602&key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:** key, isbns: Array of ISBNs or a comma separated string of ISBNs (1000 ISBNs per request max.), format: json, callback: function to wrap JSON response

# **Get the reviews for a book given a Goodreads book id**

Get an XML or JSON response that contains embed code for the iframe reviews widget. The reviews widget shows an excerpt (first 300 characters) of the most popular reviews of a book for a given internal Goodreads book\_id. Reviews of all known editions of the book are included. XML responses also include shelves and book meta-data (title, author, et cetera). The Goodreads API gives you full access to Goodreads-owned meta-data, but it does not give you full access to book meta-data supplied by third parties such as Ingram. Book cover images, descriptions, and other data from third party sources might be excluded, because we do not have a license to distribute these data via our API. If you need book meta-data beyond what our API provides, consider signing up for an Amazon developer key.  
**URL:** [https://www.goodreads.com/book/show.FORMAT](https://www.goodreads.com/book/show/50.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/book/show/50.xml?key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:** key format: xml or json, id: A Goodreads internal book\_id, text\_only: Only show reviews that have text (default false), rating: Show only reviews with a particular rating (optional)

# **Get the reviews for a book given an ISBN**

Get an xml or json response that contains embed code for the iframe reviews widget that shows excerpts (first 300 characters) of the most popular reviews of a book for a given ISBN. The reviews are from all known editions of the book.  
**URL:** [https://www.goodreads.com/book/isbn/ISBN?format=FORMAT](https://www.goodreads.com/book/isbn/0441172717?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/book/isbn/0441172717?key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET  
**Parameters:**

* format: xml or json
* callback: function to wrap JSON response if format=json
* key: [Developer key](https://www.goodreads.com/api/keys) (required only for XML).
* user\_id: 18207016 (required only for JSON)
* isbn: The ISBN of the book to lookup.
* rating: Show only reviews with a particular rating (optional)

**Example code for using json with callback:**

<script type="text/javascript">

function myCallback(result) {

alert('nb of reviews for book: ' + result.reviews.length);

}

var scriptTag = document.createElement('script');

scriptTag.src = "https://www.goodreads.com/book/isbn/0441172717?callback=myCallback&format=json&user\_id=18207016";

document.getElementsByTagName('head')[0].appendChild(scriptTag);

</script>

# **Get the reviews for a book given a title string**

Get an xml response that contains embed code for the iframe reviews widget, which shows an excerpt (first 300 characters) of the most popular reviews of a book for a given title/author. The book shown will be the most popular book that matches all the keywords in the input string. The reviews are from all known editions of the book.  
**URL:** [https://www.goodreads.com/book/title.FORMAT](https://www.goodreads.com/book/title.xml?author=Arthur+Conan+Doyle&key=TNY6fDFILUDCTXZ739z56w&title=Hound+of+the+Baskervilles)    [(sample url)](https://www.goodreads.com/book/title.xml?author=Arthur+Conan+Doyle&key=TNY6fDFILUDCTXZ739z56w&title=Hound+of+the+Baskervilles)  
**HTTP method:** GET  
**Parameters:**

* format: xml or json
* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* title: The title of the book to lookup.
* author: The author name of the book to lookup. This is optional, but is recommended for accuracy.
* rating: Show only reviews with a particular rating (optional)

# **Follow a user**

Start following a user [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/user/USER\_ID/followers?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* USER\_ID: Goodreads user id of user to follow

# **Unfollow a user**

Stop following a user [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/user/USER\_ID/followers/stop\_following.xml](https://www.goodreads.com/api/index)  
**HTTP method:** DELETE  
**Parameters:**

* USER\_ID: Goodreads user id of the user you want to stop following

# **Confirm or decline a friend recommendation**

Confirm or decline a friend recommendation for the current user [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/friend/confirm\_recommendation.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* id: friend recommendation id
* response: Y or N

# **Confirm or decline a friend request**

Confirm or decline a friend request for the current user [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/friend/confirm\_request.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* id: friend request id
* response: Y or N

# **Get friend requests**

Returns a XML with the current user's friend requests '[using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/friend/requests.xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET  
**Parameters:**

* page: 1-N page of results to show (optional, default 1)

# **Add a friend**

Sends a friend request to a user [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/friend/add\_as\_friend.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* id: Goodreads user id for friend

# **Join a group**

Let the current user join a given group [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/group/join?format=xml&id=GROUP\_ID](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* id: id of the group (required)
* format: xml (required)

# **List groups for a given user**

Returns xml list of groups the user specified by id belongs to  
**URL:** [https://www.goodreads.com/group/list/USER\_ID.xml](https://www.goodreads.com/group/list/18207016.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/group/list/18207016.xml?key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET  
**Parameters:**

* sort: One of 'my\_activity', 'members', 'last\_activity', 'title' ('members' will sort by number of members in the group)
* key: [Developer key](https://www.goodreads.com/api/keys) (required).

# **Return members of a particular group**

Returns an XML list of members of the group  
**URL:** [https://www.goodreads.com/group/members/GROUP\_ID.xml](https://www.goodreads.com/group/members/1.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/group/members/1.xml?key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET  
**Parameters:**

* sort: One of 'last\_online', 'num\_comments', 'date\_joined', 'num\_books', 'first\_name'
* q: List of names to search for, separating each name with a space character. Optional, will find all members by default
* page: Which page of results to show (default 1)
* key: [Developer key](https://www.goodreads.com/api/keys) (required).

# **See the current user's notifications**

Viewing any new notifications here will mark them as 'viewed'. [using OAuth](https://www.goodreads.com/api/documentation#oauth)  
**URL:** [/notifications.xml](https://www.goodreads.com/notifications.xml)    [(sample url)](https://www.goodreads.com/notifications.xml)  
**HTTP method:** GET  
**Parameters:**

* page: 1-N page number (optional, default 1)

# **Add to books owned**

Adds a book to user's list of owned books [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/owned\_books.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* owned\_book[book\_id]: id of the book (required)
* owned\_book[condition\_code]: one of 10 (brand new), 20 (like new), 30 (very good), 40 (good), 50 (acceptable), 60 (poor)
* owned\_book[condition\_description]: description of book's condition
* owned\_book[original\_purchase\_date]: when book was purchased
* owned\_book[original\_purchase\_location]: where this book was purchased
* owned\_book[unique\_code]: BookCrossing id (BCID)

# **List books owned by a user**

Get an xml response with a list of owned books [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/owned\_books/user?format=xml](https://www.goodreads.com/owned_books/user?format=xml&id=18207016)    [(sample url)](https://www.goodreads.com/owned_books/user?format=xml&id=18207016)  
**HTTP method:** GET  
**Parameters:**

* id: Goodreads user\_id
* page: 1-N (optional, default 1)

# **Show an owned book**

Get an xml response describing an owned book [using OAuth](https://www.goodreads.com/api/documentation#oauth), including the current owner's user id (current\_owner\_id), it's BookCrossing ID (bcid), etc. OWNED\_BOOK\_ID is a unique identifier for the owned book (not a book\_id) You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/owned\_books/show/OWNED\_BOOK\_ID?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET

# **Update an owned book**

Updates a book a user owns [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/owned\_books/update/ID?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** PUT  
**Parameters:**

* id: id of the owned book record
* owned\_book[book\_id]: id of the book (required)
* owned\_book[condition\_code]: one of 10 (brand new), 20 (like new), 30 (very good), 40 (good), 50 (acceptable), 60 (poor), (optional)
* owned\_book[condition\_description]: description of book's condition (optional)
* owned\_book[original\_purchase\_date]: when book was purchased (optional)
* owned\_book[original\_purchase\_location]: where this book was purchased (optional)
* owned\_book[unique\_code]: BookCrossing id (BCID) (optional)

# **Delete an owned book**

Deletes a book a user owns [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/owned\_books/destroy/ID?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* id: id of the owned book record

# **Get a user's read status**

Get information about a read status update.  
**URL:** [https://www.goodreads.com/read\_statuses/ID?format=xml&key=KEY](https://www.goodreads.com/api/index)  
**HTTP method:** GET  
**Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* id: read status id

# **Get a recommendation from a user to another user**

Get information about a particular recommendation that one user made for another [using OAuth](https://www.goodreads.com/api/documentation#oauth). Includes comments and likes. You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/recommendations/ID?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET  
**Parameters:**

* id: recomendation id

# **Add review**

Add book reviews for members [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/review.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* book\_id: Goodreads book\_id (required)
* review[review]: Text of the review (optional)
* review[rating]: Rating (0-5) (optional, default is 0 (No rating))
* review[read\_at]: Date (YYYY-MM-DD format, e.g. 2008-02-01) (optional)
* shelf: read|currently-reading|to-read|[<USER SHELF NAME>](https://www.goodreads.com/api/index#user_shelves.create) (optional, must exist, see: [shelves.list](https://www.goodreads.com/api/index" \l "shelves.list))

# **Edit a review**

Edit a book review [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required). This method should be called with a PUT request but we support POST as well.  
**URL:** [https://www.goodreads.com/review/REVIEW\_ID.xml](https://www.goodreads.com/api/index)  
**HTTP method:** PUT  
**Parameters:**

* id: Review Id
* review[review]: Text of the review (optional)
* review[rating]: Rating (0-5) (optional, default is 0 (No rating))
* review[read\_at]: Date (YYYY-MM-DD format, e.g. 2008-02-01) (optional)
* finished: true to mark finished reading (optional)
* shelf: read|currently-reading|to-read|[<USER SHELF NAME>](https://www.goodreads.com/api/index#user_shelves.create) (optional, must exist, see: [shelves.list](https://www.goodreads.com/api/index" \l "shelves.list))

# **Delete a book review**

Deletes a book review [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required). This method should be called with a DELETE request but we support POST as well.  
**URL:** [https://www.goodreads.com/review/destroy/12345?format=xml](https://www.goodreads.com/api/index)  
**HTTP method:** DELETE  
**Parameters:**

* id: Review Id

# **Get the books on a members shelf**

Get the books on a members shelf. Customize the feed with the below variables. Viewing members with profiles who have set them as visible to members only or just their friends requires [using OAuth](https://www.goodreads.com/api/documentation#oauth).  
**URL:** [https://www.goodreads.com/review/list?v=2](https://www.goodreads.com/review/list/18207016.xml?key=TNY6fDFILUDCTXZ739z56w&v=2)    [(sample url)](https://www.goodreads.com/review/list/18207016.xml?key=TNY6fDFILUDCTXZ739z56w&v=2)  
**HTTP method:** GET  
**Parameters:**

* v: 2
* id: Goodreads id of the user
* shelf: read, currently-reading, to-read, etc. (optional)
* sort: title, author, cover, rating, year\_pub, date\_pub, date\_pub\_edition, date\_started, date\_read, date\_updated, date\_added, recommender, avg\_rating, num\_ratings, review, read\_count, votes, random, comments, notes, isbn, isbn13, asin, num\_pages, format, position, shelves, owned, date\_purchased, purchase\_location, condition (optional)
* search[query]: query text to match against member's books (optional)
* order: a, d (optional)
* page: 1-N (optional)
* per\_page: 1-200 (optional)
* key: [Developer key](https://www.goodreads.com/api/keys) (required).

# **Recent reviews from all members (xml)** **URL:** [https://www.goodreads.com/review/recent\_reviews.xml](https://www.goodreads.com/review/recent_reviews.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/review/recent_reviews.xml?key=TNY6fDFILUDCTXZ739z56w)

# **HTTP method:** GET **Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).

# **Get a review (xml)** **URL:** [https://www.goodreads.com/review/show.xml](https://www.goodreads.com/review/show.xml?id=22&key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/review/show.xml?id=22&key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET **Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* id: id of the review
* page: 1-N. The page number of comments. (default 1, optional)

# **Get a user's review for a given book (xml)** **URL:** [https://www.goodreads.com/review/show\_by\_user\_and\_book.xml](https://www.goodreads.com/review/show_by_user_and_book.xml?book_id=50&key=TNY6fDFILUDCTXZ739z56w&user_id=1)    [(sample url)](https://www.goodreads.com/review/show_by_user_and_book.xml?book_id=50&key=TNY6fDFILUDCTXZ739z56w&user_id=1) **HTTP method:** GET **Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* user\_id: id of the user
* book\_id: id of the book
* include\_review\_on\_work: 'true' or 'false' indicating whether to return a review for another book in the same work if review not found for the specified book (default 'false', optional)

# **Find an author by name (xml)**

**URL:** [https://www.goodreads.com/api/author\_url/<ID>](https://www.goodreads.com/api/author_url/Orson%20Scott%20Card?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/api/author_url/Orson%20Scott%20Card?key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET  
**Parameters:**

* id: Author name
* key: [Developer key](https://www.goodreads.com/api/keys) (required).

# **Find books by title, author, or ISBN**

Get an xml response with the most popular books for the given query. This will search all books in the title/author/ISBN fields and show matches, sorted by popularity on Goodreads. There will be cases where a result is shown on the Goodreads site, but not through the API. This happens when the result is an Amazon-only edition and we have to honor Amazon's terms of service.  
**URL:** [https://www.goodreads.com/search/index.xml](https://www.goodreads.com/search/index.xml?key=TNY6fDFILUDCTXZ739z56w&q=Ender%27s+Game)    [(sample url)](https://www.goodreads.com/search/index.xml?key=TNY6fDFILUDCTXZ739z56w&q=Ender%27s+Game)  
**HTTP method:** GET  
**Parameters:**

* q: The query text to match against book title, author, and ISBN fields. Supports boolean operators and phrase searching.
* page: Which page to return (default 1, optional)
* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* search[field]: Field to search, one of 'title', 'author', or 'all' (default is 'all')

# **See a series (xml)**

Info on a series  
**URL:** [https://www.goodreads.com/series/show/ID.xml](https://www.goodreads.com/series/40321-drina?format=xml&key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/series/40321-drina?format=xml&key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET

# **See all series by an author (xml)**

List of all series by an author  
**URL:** [/series/list?format=xml&id=AUTHOR\_ID](https://www.goodreads.com/series/list/227840.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/series/list/227840.xml?key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET

# **See all series a work is in** **URL:** [https://www.goodreads.com/series/work/WORK\_ID?format=xml](https://www.goodreads.com/work/118-drina-ballerina/series?format=xml&key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/work/118-drina-ballerina/series?format=xml&key=TNY6fDFILUDCTXZ739z56w) **HTTP method:** GET

# **Get your friend updates**

Get your friend updates (the same data you see on your homepage) [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/updates/friends.xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET  
**Parameters:**

* update: Type of update. Valid values are: books, reviews, statuses. (optional, default all)
* update\_filter: Which updates to show. Options are: friends (default - includes followers), following, top\_friends. (optional)
* max\_updates: The max limit of updates. (optional)

# **Add bookshelf**

Add book shelves for members [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/user\_shelves.xml](https://www.goodreads.com/api/index)  
**HTTP method:** POST  
**Parameters:**

* user\_shelf[name]: Name of the new shelf
* user\_shelf[exclusive\_flag]: 'true' or 'false' (optional, default false)
* user\_shelf[sortable\_flag]: 'true' or 'false' (optional, default false)
* user\_shelf[featured]: 'true' or 'false' (optional, default false)

# **Edit bookshelf**

Edit a shelf [using OAuth](https://www.goodreads.com/api/documentation#oauth). You'll need to [register your app](https://www.goodreads.com/api/keys) (required).  
**URL:** [https://www.goodreads.com/user\_shelves/USER\_SHELF\_ID.xml](https://www.goodreads.com/api/index)  
**HTTP method:** PUT  
**Parameters:**

* user\_shelf[name]: Name of the new shelf
* user\_shelf[exclusive\_flag]: 'true' or 'false' (optional, default false)
* user\_shelf[sortable\_flag]: 'true' or 'false' (optional, default false)
* user\_shelf[featured]: 'true' or 'false' (optional, default false)

# **Get info about a member by id or username**

Get an xml response with the public information about the given Goodreads user.  
**URL:** [https://www.goodreads.com/user/show/18207016.xml](https://www.goodreads.com/user/show/18207016.xml?key=TNY6fDFILUDCTXZ739z56w)    [(sample url)](https://www.goodreads.com/user/show/18207016.xml?key=TNY6fDFILUDCTXZ739z56w)  
**HTTP method:** GET  
**Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* id: Goodreads user id.
* username: Goodreads user name (not first name). Usernames are optional on Goodreads.

# **Get a user status**

**URL:** [https://www.goodreads.com/user\_status/show/ID?format=xml&key=KEY](https://www.goodreads.com/api/index)  
**HTTP method:** GET  
**Parameters:**

* key: [Developer key](https://www.goodreads.com/api/keys) (required).
* id: user status id

# **View user statuses**

View most recent user statuses on the site.  
**URL:** [https://www.goodreads.com/user\_status/index.xml](https://www.goodreads.com/api/index)  
**HTTP method:** GET

Not used yet:

Create comment, Events in your area, get info about a topic with ID, get a list of topics with unread comments, get a list of topics in a group’s folder, create a new topic, add a quote, like a resource, unlike a resource, find a group, get info about a group by id, return members of a particular group, join a group, create a comment, list comments on a subject, listopia lists