

# Using the Meetup API with Yahoo! Pipes

## *White Paper on NassauWalks.org*

*(preliminary draft, rev. 0.1)*

© 2008, Beads Land

beads@nassauwalks.org

## Preamble and Contents

This document represents a first attempt at getting the author's thoughts and experience down on paper. Atrociously written, haphazardly organized, and out of date before it was even finished, the text that follows is at best a source of raw material from which a far better document might be assembled. The reader is asked to forgive the author's baroque writing style (at once both verbose and dense), and assured that significant revisions are anticipated (constructive feedback, therefore, is encouraged).

This mini-tome weaves together discussion of four underlying themes, each of which might easily deserve a paper of its own:

- **The Meetup API** – released with the redesign of the Meetup.com Web site in June 2008, the Meetup API offers developers the opportunity to significantly extend the reach and power of the Meetup.com platform. That said, the Meetup API service still needs a lot of work to bring it up to its full potential.
- **Yahoo! Pipes** – launched in 2007, the Yahoo! Pipes platform affords powerful tools for building and integrating data feeds for use in Web applications. It too, however, has areas that could be improved upon.
- **The Digital Divide** – a soapbox issue of the author (be prepared for pedantry), this is of specific concern when an Web-based service (namely, Meetup.com) aims to organize people in the real world. The real world is populated by people representing a wide range of Internet savvy. In this author's experience, many people would rather try to find their way around town on foot than try to find their way around a Web site.<sup>1</sup>
- **Simplicity Through Complexity** – this theme is still to be fully articulated. Essentially, the premise is that highly-technical multi-purpose systems can and should be used to build easy-to-understand, single-purpose solutions. This paper gets very deeply into the highly-technical complexity... the easy-to-understand bit is still wanting.

---

<sup>1</sup> As your author organizes a group devoted to walking, this is actually a preference to be encouraged. That said, there are many predisposed to engage in neither activity—who might still make great contributions to the extended Meetup.com community, but for the hurdle of getting around on the Web.

## Table of Contents

Introduction.....	3
Meetup.com.....	3
Yahoo! Pipes.....	4
Nassau County Walking Meetup Group.....	5
Making the Most of Meetups.....	5
Project History.....	5
Phase 0: Problem Identification.....	6
Alternative Calendar.....	6
Easy Entry Point.....	7
Collating Sources.....	8
Audience-Orientation.....	8
Phase 1: Proof of Concept.....	10
Phase 2: Prototype.....	12
Phase 3: Production.....	13
Phase 4: Pending.....	13
Notation Used Herein.....	14
Components.....	15
Implementation.....	16
Member Count.....	16
[pageMeetupAPI].....	17
[getMeetupGroup].....	17
[walk member count].....	18
<nassauwalks.org>.....	18
Nassau's Largest.....	18
[getCountyByZip].....	19
[walk nassau meetup groups].....	19
Member Places.....	19
*** Validity of Member Place Data ***.....	20
[walk member places].....	21
Largest Walking.....	21
[walk groups].....	21
Album Thumbnails.....	22
[walk photos].....	22
Walking Quotes.....	22
[getMeetupNewBios].....	23
[getMeetupGroupMemberReviews].....	23
[getMeetupGroupMemberPhoto].....	24
[Quote Builder].....	24
[walk quotes]   <tinyurl.com>.....	24
quote.cgi.....	24
Local Calendar.....	25
Overview.....	26
*** Volume and Scale ***.....	27

[parseMeetupEventUrl].....28

[getMeetupEventType].....28

[getMeetupEventGeo].....29

\*\*\* On the Issue of Event Geodata \*\*\* .....29

[getRichMeetupGroupEvents].....31

[callDistanceJsonUrl].....32

[Proper Case].....33

[Nassau County Walking Meetup Events].....33

[recurMonthlyWalks].....35

[getMonthDayOfWeeks].....35

[getMonthOrdinalDayOfWeek].....35

[testInFloatingDateRange].....36

[floatToUSAwDST] .....36

[recurWeeklyWalks].....36

[oldWestburyWalks].....37

[muttontownWalks].....37

[walks not on meetup].....37

[discoverWalks].....37

[NassauWalks Calendar].....37

[Where We're Walking (Text)] .....38

rss.cgi .....38

<nassauwalks.org>.....38

Where We're Walking.....38

Future Development.....38

    Technical.....38

    Functional.....39

    Content.....39

    Geographic.....39

    Topics.....39

## Introduction

The Web site of the Nassau County Walking Meetup Group was conceived, initially, with a dual purpose: first, as proof of concept of customized Meetup group site design; second, to provide a meaningful entry point for prospective members. Yahoo! Pipes proved a powerful platform for seamlessly transforming RSS<sup>2</sup> feeds and distilling public Web pages, and thereby feeding multiple features of the nassauwalks.org home page. The site has evolved into a local topic-oriented events calendar portal, and, with the conversion to Meetup's new API<sup>3</sup>, nassauwalks.org is poised to become a

---

2 Varies, Really Simple Syndication, RDF Site Summary, or Rich Site Summary. *RSS* provides a standardized format for publishing frequently or semi-frequently updated content for aggregation with other sources and redistribution across multiple applications. To deliver content via RSS is to *syndicate* said content, and to retrieve content regularly via RSS is to *subscribe* to same.

3 Application Programming Interface. *APIs* provide third-party developers the opportunity to work with data and functionality of a common application or platform. By exposing a standardized set of public methods, APIs serve to shield developers from implementation changes, while granting the assurance of a sanctioned means of access to the underlying application.

virtual *looking glass*<sup>4</sup>, providing local walking enthusiasts with an integrated interface for coordinating face-to-face activities across groups and venues.

This white paper<sup>5</sup> is intended to provide a case study of actual applications leveraging the power of the Meetup API and Yahoo! Pipes, to serve as a study and reference guide for the API Working Group of Making the Most of Meetups, and as a functional and technical specification for future development of nassauwalks.org and related applications and Web sites. Where relevant, suggestions for improvement are also provided throughout.

All that said, this document is presently only in preliminary draft form. Future revisions will be available from the Nassau County Walking Meetup Group Web site: <http://www.nassauwalks.org/>

## **Meetup.com**

Founded in 2002, Meetup.com defies the usual categories of Web-facilitated communities. Unlike sites that fit neatly under the label of “social networking”, the focus of Meetup is on bringing people together, face-to-face, in the real world. Yet, despite frequently being confused with “online dating services”—in many people's minds, the only Web sites with a purpose of having people “meet up” in person—Meetup is focused not on romance<sup>6</sup>, but on social and civic involvement: social networks not as an end in themselves, but as the stuff of social capital<sup>7</sup>, the realization of freedom of association as public good. With nearly 46,500 groups, each lead by a volunteer “organizer”, across 112 countries, spanning over 3,600 topics of interest, Meetup defines a new category onto itself.

## **Yahoo! Pipes**

Founded in 2007, Yahoo! Pipes stands ready to realize the true power of the much hyped “Web 2.0”<sup>8</sup>. Users of Pipes employ a visual editor to assemble and configure chains of modules that retrieve,

---

4 See Future Development section, at the end of this paper, for more discussion of the *looking glass*.

5 Historically referring to a document setting forth the policy of a government body, the term *white paper*, has, in the last decade or so, come to be used for quasi-technical documents that educate readers about how given technologies can be or have been used to solve practical problems of interest. Part case study, part specification, a white paper is generally used to make a case for the benefits of a subject technology. Admittedly, this early draft document has a long way to go before it will effectively accomplish such a task. High aspirations.

6 Member profiles on Meetup.com do not even include a field for gender, such that Meetup organizers have long been familiar with the formulaic e-mail subject line “So-and-so has changed his or her RSVP for such-and-such”—although, recently, the grammatically-correct, if somewhat awkward (and politically suspect), “his or her” has been foreshortened to more ambiguous, if less bulky, “their”. This is not to say that the Meetup.com application does not host numerous singles groups (871 at last count), only that it is not the *raison d'etre* of Meetup.

7 Like its cognates, physical capital (*e.g.*, goods, tools, raw materials) and human capital (*i.e.*, labor, also, generalized expertise and education characteristic of individuals or labor pools), *social capital* is an expression of the relationships and interactions between individuals and groups that add value and drive productivity. Unlike physical capital, which is “depleted by use” (*i.e.*, used up or worn out), the value of social capital declines when it goes unused. By corollary, the more we each relate to other human beings, the more valuable our relationships become, to ourselves, to those we relate to, and to society and the world at large.

8 If we retroactively define “Web 1.0” as encompassing one-way publishing on Web pages and Web sites, whether static or dynamic in nature, *Web 2.0*, at its most basic, introduces two-way publishing—although usually still in the context of a single Web site (*e.g.*, feedback left by readers of a blog). The potential of Web 2.0, however, hides in the often neglected RSS feeds, commonly backwater REST APIs, and largely unknown “mashups”, which extend the reach of applications across Web sites and organizational boundaries.

process, and transform sets of data, information, and content. The resulting data mashups<sup>9</sup> can then be exposed as Web feeds<sup>10</sup>, embedded in Web pages, incorporated into server-based Web applications, and/or chained into other Pipes applications. Although most Pipes applications tend to focus on aggregation (*i.e.*, drawing together several sources of data into a single feed), attrition (*i.e.*, filtering items in a feed, based on predetermined criteria), and/or augmentation (*i.e.*, introducing new elements into items of a feed<sup>11</sup>), Pipes is a fully Turing-complete<sup>12</sup> visual language, capable of generating as well as consuming and manipulating data.

## **Nassau County Walking Meetup Group**

Founded in 2007, the Nassau County Walking Meetup Group brings together walking enthusiasts from Nassau County, New York and the surrounding area for fitness, to relieve stress, get out into nature, explore our neighborhoods, meet new people, and maybe just make a difference. The third largest Meetup group in Nassau County<sup>13</sup>, and the third largest walking Meetup group in New York State<sup>14</sup>, the Nassau County Walking Meetup Group represents nearly 300 members and has held over sixty events to date. The group's Web site, [nassauwalks.org](http://nassauwalks.org), provides a calendar of local walking events, both those scheduled by Meetup organizers and those held by a number of local pre-Meetup organizations and

- 
- 9 A term originating in the arts, *mashup* refers to the mixing together of elements of two more exiting pieces to derive something new. Like collage and *décollage* in traditional media, musical, video, and digital mashups are constructed by building up and tearing away parts of source works. Application mashups use the same processes to build up, slice and dice, and transform data, information, and content for presentation to end users. However, unlike works of art, which typically are fixed in some static form, application mashups tend to draw from dynamic sources, such that content is refreshed and updated as elements are added to, removed from, and modified in the underlying pieces.
  - 10 As in the telecommunications industry before it, *feeds* on the Web allow consumers of data, information or content to receive updates over time. Pre-Internet examples of feeds include paper-tape stock tickers and television networks' transmission of programming to affiliated stations. Web feeds, more often than not, involve content organized by date of publication, such as news, blog postings, and podcast episodes.
  - 11 A special case of augmentation is the creation of GeoRSS feeds, wherein latitude and longitude coordinates are added to feed items. Such a resulting feed is displayed to end users not as a list (chronologically sorted or otherwise), but rather as a map, with each feed item represented by a marker or image.
  - 12 *Turing completeness* refers to the ability of a system to perform all the computations of a universal Turing machine, and thus, to perform any computer algorithm. Although other programming environments may be better adapted to certain programming tasks (*e.g.*, the great-circle distance calculation is probably best performed in a context that provides trigonometric functions natively), the potential of Yahoo! Pipes extends well beyond the combination and transformation of data feeds.
  - 13 Largest Meetup groups in Nassau County, New York: (1) Long Island NY Middleaged & Young-at-Heart Singles Meetup Gp; (2) Motherhood Later Than Sooner: 35+ Moms in NY; (3) Nassau County Walking Meetup Group; (4) Dinners, movies, and etcetera for singles; (5) Nassau County Brunch Friends Meetup Group. As of July 23, 2008, there are 156 Meetup groups in Nassau County. That two singles-oriented Meetup groups are among the top five largest in Nassau County arguably evidences the conflation of Meetup.com with online dating (as discussed earlier in this paper). The full scope of Meetup.com's potential to bring together people across a wide range of interests has yet to be realized in Nassau County and on Long Island generally.
  - 14 Largest walking Meetup groups (where "walkers" is the primary category of the group) in New York State: (1) The New York Walking Meetup Group, (2) Athletic Fitness & Social Walkers; (3) Nassau County Walking Meetup Group; (4) The Rochester Hiking Meetup Group; (5) The NYC Broadway Walk Meetup. As of July 23, 2008, there are 12 walking Meetup groups in New York State, out of approximately 178 walking Meetup groups world-wide (the number fluctuates from day to day). All 12 walking Meetup groups in the state are located in the New York City metropolitan area.

institutions: encouraging members and prospective members to participate in walking events that most appeal to them.

## ***Making the Most of Meetups***

Founded in the Spring of 2008, Making the Most of Meetups is the premier<sup>15</sup> Meetup organizer Meetup group for Nassau County, Queens, and western Suffolk County<sup>16</sup>. Part book club (reading, for starters, the books that inspired the creators of Meetup.com), part Web technology group, Making the Most of Meetups conducts workshops and discussion forums exploring the potential of Meetup.com and the techniques that Meetup members—organizers and non-organizers alike—can use to build great groups, both to satisfy individual desire for more and/or deeper social connections, and to create positive experiences for fellow members that enhance the value of groups for everyone. Intended to serve a wide range of members, regardless of technological savvy, future events will run the gamut from basic Internet e-mail and Web skills to Meetup API development and group Web site design, as well as communications and event planning skills development, and discussions on topics ranging from sociology to evolution.

## **Project History**

The history of the [nassauwalks.org](http://nassauwalks.org) Web site can be discussed along a number of phases of development: (i) a problem identification stage, during which the initial purposes of the Web site were identified; (ii) an initial proof of concept, demonstrating what an API for Meetup.com might enable, (iii) a prototype of an actual API-driven Web site; (iv) a production version of the same site, constructed from reusable components; and (v) a pending development list. We review each of these stages of the site's development here.

### ***Phase 0: Problem Identification***

The Web site of the Nassau County Walking Meetup Group was developed for with purposes: (i) as an alternative to a manually-edited monthly group calendar/newsletter; (ii) as a easy-to-relate (and easy-to-remember) entry point for prospective members; (iii) to streamline the process of collating walking event data from multiple sources; and (iv) as a proof-of-concept of how the Meetup.com engine could facilitate the delivery of audience-specific content, were an API available.

## **Alternative Calendar**

From shortly after the launch of the Nassau County Walking Meetup Group, your humble organizer has made use of the notes function of the group's calendar<sup>17</sup> on Meetup.com to track and advise members of

---

15 There are three other Meetup organizer Meetup groups in the same region, one each in Nassau, Queens, and Suffolk counties, each specializing in Meetup moms organizers, and two of which are private.

16 Already the sixth largest Meetup organizer Meetup group in the State of New York, Making the Most of Meetups also draws members from nearby New York City, attracted by the depth and scope of our events.

17 The calendar feature of Meetup.com recognizes two types of item: “meetings” and “notes”. Meetings are events held by a given Meetup group. Organizers can set a range of properties for each meeting, including details about the venue of the event, RSVP and payment policies, and even an event-specific photo. Members can RSVP for meetings and leave comments (both RSVPs and comments are displayed for the edification of other members), and once an event has

walks scheduled throughout the Nassau County area, including those held by various non-Meetup organizations. The intent being to encourage members to walk at those venues and during those times most convenient and attractive to them.

In conjunction with the calendar on Meetup, a monthly e-mail was sent to members summarizing the events listed on the calendar. For many members of the Nassau County Walking Meetup Group—those having limited experience and/or comfort with Internet applications<sup>18</sup>—an e-mail pushed to their personal e-mail box is easier to access and utilize than the Web-based calendar provided by Meetup.com. Others, although confident about their ability to find and use the calendar, expressed concerns about its limited layout and resulting difficulty of use.<sup>19</sup>

The newsletter e-mails were created on a monthly basis from April through November of 2007, with copies posted to the group's message board<sup>20</sup>. As the number of nearby walking events identified by the organizer, and listed on our Meetup group's calendar, grew however, it became increasingly time consuming to prepare these newsletters. Also, an e-mail calendar published at the beginning of the month was necessarily incomplete, as new events in the later part of the month were inevitably discovered after the e-mail edition was released. An alternative was needed.

## Easy Entry Point

A common experience with many Meetup groups, particularly those that meet in public places, is questions from passers-by, curious about a group of people gathered and engrossed in whatever activity the group may be engaged in. This may be even more frequent an occurrence for outdoors-activity oriented groups, which tend to draw attention to themselves simply by walking together as a cohesive unit. The Nassau County Walking Meetup Group, conducting many of its walks on the sidewalks of Long Island's post-war suburban developments (characterized, as they are, by a strong emphasis on travel by personal automobile), is especially prone to such questions.

---

occurred, those that attended can rate and review their experience. Notes, by contrast, are far less functional, but still useful, static text reminders of events the group organizer chooses to share with members.

- 18 The Internet-savvy among us sometimes find it hard to imagine, but there is nothing intuitive about navigating the Web, whether finding your way around a Web site, finding your way back to a Web site once visited, or even finding your way into a Web browser on a modern personal computer. These are learned skills, like any others. We should always make an effort to adapt the technology to the user, and not expect the user (or potential user) to adapt to our preferred technology. Web sites, unlike ball fields, do not come with a “If you build it, they will come” guarantee.
- 19 Prior to the recent upgrade of Meetup.com's Web site, group calendars presented the month in a wall calendar format, with each box for each day. (There was, and still is, an alternative view of the calendar, in list form—but many members have had difficulty finding this form of the calendar on the site.) The boxes were of fixed height, such that if multiple meetings and/or notes were scheduled for the same day, a scroll bar would appear in the box for that day. In the new site design, the boxes instead stretch as necessary to accommodate all of the events listed for a given day. Needless to say, this recently implemented change greatly aids in reading the calendar.
- 20 The message board feature of Meetup.com, once standard (and now optional—organizers may turn it on or off) for all Meetup groups, is a threaded discussion forum, a common staple of online communities. Unlike a mailing list, where each message posted to the list is copied by e-mail to all other members—either singly or in digest form—a discussion forum exists entirely on a Web site: users must visit the Web site to read new and respond to new posts. Discussion forum applications are commonly *threaded*, containing a number of message threads, each starting with an original post—labeled with the subject line of said post—followed by all of the responses to said post, and the responses to those responses, in chronological order. Message boards are just one some half a dozen different ways in which Meetup members can communicate with others through the site, including messages, greetings (previously called “shouts”), RSVP comments, event comments, event reviews, and (optional) mailing lists.

On learning about the group, and about Meetup.com, the next natural question might be “How do I learn more?” or “How do I join?” This typically leads to an answer along the lines of “Well, first you go to [meetup.com](http://meetup.com), click the button to find a Meetup, enter your zip code and search<sup>21</sup> for walking, and then... and then....” Curiosity goes the way of glazed eyes.

An added complexity is that those unfamiliar with Meetup may struggle to disambiguate the Meetup group from Meetup itself. “So, are you Meetup?” is a question often posed by those who's first encounter with the Meetup logo is in the form of a sign held or button worn by the organizer of a Meetup group. In this light, providing a prospective member with the URL of a Meetup group, “*topic.meetup.com/number*”, whether orally or on printed matter, may leave ambiguous the relationship between particular group and the Web site that hosts it.

That such formulaic URLs (complete with their arbitrarily assigned group number) are easy to forget is an added hurdle. Printed materials may overcome the memorization hurdle, but only if they are not discarded, misplaced, inadvertently laundered, or hopelessly buried among the fibrous miscellanea characteristic of many a wallet, handbag, jacket pocket, passenger-side seating area, and the various domestic stratified accretions of paper upon which the contents of such conveyances are later deposited. This, additionally, all assumes the printed matter in question is not a T-shirt worn by a group member—where again, whether the URL is easily remembered (*i.e.*, mnemonically meaningful) becomes paramount.<sup>22</sup>

## Collating Sources

As mentioned above, what started out as an effort to track and notify group members of other walking events occurring in the Nassau County area, soon became a time consuming endeavor, as more and more walks were identified throughout the area: pre-Meetup walking and hiking<sup>23</sup> organizations, state and county parks, charitable fundraising organizations, historical societies, nature preserves, Audubon chapters, in addition to newly founded Meetup groups, quickly filled out the group's calendar.

The good news was that the vast majority of this information was available online. Indeed, in an effort to ensure that members had access to the most up-to-date information about walking related events, the decision was made to only list events on the group's calendar if a link could be included to a Web page

21 The search engine on Meetup.com also leaves a lot to be desired, often returning unanticipated and oddly incomplete results in response to queries. The organizer of one large walking Meetup group recently sent an e-mail to members advising that searching for their group by name failed to find the group, and requesting that members contact Meetup.com support to request that the problem be fixed.

22 It should be noted, at this juncture, that the provision of an easy-to-remember URL does not require the construction of a Web site. A domain name registered for a \$20 per year fee can easily be set up to direct users to a Meetup group's home page on Meetup.com. See, for example, <http://www.mojoemineola.com/>

23 Although the terms are commonly used interchangeably, here we make a distinction between *walking*, or ambulation generally, and *hiking*, which more typically refers to walking in natural contexts. Although the term *hiking* is also sometimes used to refer to walking long distances, the inherent ambiguity of this usage (what constitutes a “long” distance is a matter of opinion), and the potential discouraging connotations that may attach (where “long” implies hard, or unduly challenging)—when our goal is to encourage walking (for any and all distances)—mitigates against this usage for our purposes. Rather, we shall consider “walks”, of whatever type and in whatever setting, to be a superclass, to which “hikes” is a subclass, referring to walks in undeveloped (or reclaimed/restored) settings, usually along trails or paths maintained by hiking enthusiasts, and/or authorized bodies, for that purpose. That hikes may also involve special equipment and/or preparation, or occur in out-of-the-way places, further distinguishes this subcategory from more spontaneous and unencumbered varieties of walking.

describing the event. What was needed, then, was a way to automate the processing and listing of these events, so that members and casual visitors would have a consistently up-to-date and comprehensive calendar of walking events at their fingertips—while allowing the organizer more time to organize and promote the group's own walking events.

## Audience-Orientation

Meetup.com is a powerful platform for organizing groups that bring people together in meaningful and productive ways. Focusing as it does on providing an architecture that facilitates and fosters the self-organizing of face-to-face groups, the Meetup.com Web site necessarily is designed at a level of abstraction that can be disorienting for the uninitiated. This is particularly the case for those for whom the Internet is still a new and unfamiliar technology.

Coming as it does from a post-Internet perspective, Meetup.com presents itself as the next step for those experienced Internet users who, having tried to find people they share interests with online, can now “Use the Internet to get off the Internet.” This messaging assumes an audience of Facebook-jockey, weblog-junkie, Myspace-escapee, flame-war veterans who, having grown up with the Internet as the pervasive firmament of ultra-modern life, are now open to finding out what the wider world may have to offer.

For the vast majority of the population<sup>24</sup>, for whom the Internet is still a newfangled, perhaps fascinating, certainly a little unsettling, reflection of a mysterious subculture, who know more about the Web from stories on the evening news than from direct experience, and, if they do own a computer, may be inclined to make about as much use of it as they do the equally cryptic “food processor” stowed away on the top shelf of the pantry closet (and, indeed, would probably stow the computer away much the same way, if only it were as compact as other little-used dust-gathering home appliances), the “been to that blog, done that social network” orientation is as opaque and alien as the place of parenthood is to someone who has never held an infant.

The expectations held by Meetup.com are evident in the site's spare, functional design. Leveraging now well-established (for long-term Web users, at least) visual metaphors and page layout for site navigation, and deploying a site architecture of functionally-organized, hierarchically-structured content, the Meetup.com site assumes a certain level of computer literacy. For those who attained *literacy* in a world of newspapers, magazines, and books, the placement of important information on a Web page—and the weird ways in which certain textual elements serve functional rather than contextual aims—can be disorienting to the point of incomprehension. For those who are baffled by the nephew reading his manga comic (ahem, graphic novel) backwards, finding one's way around a Web site can be like learning to read all over again.<sup>25</sup>

24 As of 2008, it is estimated that approximately 72% of the U.S. population fit in the category of “Internet users”. This, however, is likely a measure of those who answered positively to a question of the form “Have you ever gone online?” This is the equivalent of asking “Have you ever played a sport?” and then using the results to indicate how many athletes there are in a population. Today, it is not uncommon for a consumer to have Internet access at home (presuming, first, that they have a computer at home), but to use it only sparingly, or to limit their usage to the neighborhood of content bundled with access by their Internet Service Provider. Many who use the Internet at work, likewise, are familiar enough with the functionality of a few Web sites (having learned how to use them much as one learns how to use a new software application on the desktop), but become easily frustrated when confronted with new Web sites or Web sites they use only infrequently.

25 Indeed, a common refrain among of those in the *technoliterati*, upon encountering a novice Internet user as they struggle

Now, drop into this milieu a novice Internet user trying to find a walking group, the members of whom they met only a few days ago in the middle of town. The red-and-white-and-pink color scheme hardly communicates “walking” (or “hiking” or “nature” or any of the other associations one might make with a group dedicated to walking), and the busy front page of the site, with its embedded YouTube video and ever-changing map of recent event RSVPs across the continental United States<sup>26</sup>, can likewise be puzzling to someone who thought they were going to learn about a local walking group.

---

The solution: a Web site that presents the information prospective members are looking for, using a straight-forward, top-to-bottom layout. Presenting a detailed calendar of local walking events<sup>27</sup>, without the necessity of searching, and postponing the process of joining the group and RSVPing for events until after the reader has been able to peruse the events available—from Meetup groups and other organizations—the [nassauwalks.org](http://nassauwalks.org) site is focused on serving an audience interested in walking in and near Nassau County.

This interest in walking locally, of course, may translate into an interest in Meetup.com and other local Meetup groups. We will explore this further in the Future Development section of this white paper.

### **Phase 1: Proof of Concept**

The Web site of the Nassau County Walking Meetup Group grew out of work done to develop the current iteration of the personal home page of your humble author: [beadsland.com](http://beadsland.com). The [beadsland.com](http://beadsland.com) site aggregates the RSS calendars and RSS “What’s New” feeds<sup>28</sup> of the Meetup groups organized by said individual, with certain other sources of syndicated content.

After experimenting, with little satisfaction, with a number of server-side tools and Web services for aggregating syndicated content for display on Web sites, your organizer discovered Yahoo! Pipes. The ease by which the Pipes platform could be used to assemble and deploy aggregated content for use in Web pages inspired the idea of creating a walking-specific Web site using the same tools.

Thereafter, in March 2008, the [nassauwalks.org](http://nassauwalks.org) Web site was launched. Initially presenting the RSS feed of the meetings and notes appearing on the calendar of Nassau County Walking Meetup Group

---

to use and understand the strange geography of the Internet, is “Learn to read!”. On its face, a signal of the empathy challenged, this frustrated outcry brings to the surface a very basic gap in essential skills. Just as riding a bicycle comes easily to one who has struggled past the training wheels, so reading a Web site is a no-brainer for the Internet-savvy, and too often we forget that it is a skill *learned*, and that even we didn't know how to do it at one time.

26 Add to this an overzealous drop-down topic suggestions feature in the “Find a Meetup” box on the same page. On one recent visit, this helpful little bit of scripting seemed so intent on stealing the *focus* that typing a five digit number into the ZIP Code box became an exercise in frustration, with the *cursor* being redeposited repeatedly in the topic box, such that what should have been a simple matter of typing “walking” and “11590” resulted instead in a series of garbled variations on “590king”. That reloading the page resolved the issue does nothing for the inexperienced user unaccustomed to the “if it don't work, reboot it” stratagem.

27 Although not yet by e-mail. More on this later.

28 Like the calendar, the “What’s New” section of each Meetup group provides a link to subscribe to a feed of the same content. This feed is only available for public groups.

(and later, of another, newly formed, local walking Meetup group, B O A R D W A L K<sup>29</sup>), other features were soon added, eventually culminating in the feature set existing on the site today.

To supply these additional features, the proof-of-concept version of the nassauwalks.org site also drew data from the public<sup>30</sup> Web pages of both groups on Meetup.com, and from other Meetup.com pages<sup>31</sup>. This technique<sup>32</sup>, while useful in the short term for demonstrating the potential of an API-driven site, suffers from two drawbacks.

First, unlike API methods, or even RSS feeds, Web pages do not typically follow standards for delivering computer-readable data in a structured form.<sup>33</sup> As a result, a developer employing this approach must identify elements in the text to use as landmarks, by which their application can find the data to be extracted. If the page design later changes (as often happens) and these landmarks change or move in relation to one another, the data can no longer be found, and the application relying on that data fails. An API, by contrast, delivers data in the form of defined attribute/value pairings, independent of how any given Web page incorporating the same data might be designed.

Second, some Web sites frown on the use of the data on their pages in this manner. For this reason alone, being able to obtain such data via an API—where the intended purpose of the API is to make data available to developers—is preferable.

For each feature, an SSI<sup>34</sup> directive to execute a Perl<sup>35</sup> CGI<sup>36</sup> script is included in the nassauwalk.org home page<sup>37</sup>. When the home page is requested by a user, the following steps occur: (i) the Web server software parses the home page, and executes each SSI directive; (ii) the CGI script referenced by each SSI directive is executed; (iii) each given CGI script attempts to read a cache<sup>38</sup> file of Pipes feed;

29 The seventh largest walking Meetup groups in New York State, B O A R D W A L K, as its name suggests, was created to hold walks on the boardwalks of Nassau and Suffolk counties. Lee, the organizer, also organizes two other popular Long Island Meetup groups: (([ FLICK-PICK ])) and Long Island Dining Out.

30 Both groups are set to allow their events, and the venues of those events, to be viewed by non-members. This is not the case with all Meetup groups, many of which either are private (such that only the date and time of their next event is visible to non-members) or else allow non-members to view event descriptions, but only allow members to see the venues of one or more of those events.

31 For instance, the feature listing the largest walking Meetup groups in New York. The page that sourced the data used by this feature was eliminated when the Meetup.com site was redesigned, but the same information became available through the Meetup API, launched at the same time. (Subsequently, a new page, and RSS feed, providing the same data has been added to the Meetup.com site.)

32 Known, in the industry, as *scraping*. Ssssssh. Don't tell anyone.

33 Rather, Web pages primarily follow standards for denoting the structure of human-readable text.

34 Server-Side Includes. *SSI* is a simple language for embedding instructions into a Web page, for execution by the Web server software before delivery of the page to a user's Web browser—the instructions are not delivered as part of the document, but rather control the final form of the document. Here, an SSI directive is used for each CGI script, with the output of said script being embedded within the Web page itself.

35 Perl is a general purpose programming language well known to Web developers (along with its Web-oriented descendant, PHP). Actually, those familiar with Perl will probably take to Yahoo! Pipes quite easily.

36 Common Gateway Interface. CGI is a standard protocol by which programs running on a Web server are incorporated into a Web site. The “gateway” refers to the mechanisms by which the Web server software passes data and parameters into the program, and the program thereafter delivers some output (an HTML document, for instance) to the Web server.

37 In keeping with the desire to provide a simple list of upcoming events, the entirety of nassauwalks.org presently consists of a single page. While perhaps inelegant, this design does not suffer from the interminable clicks that often stymie novice users trying to find information on more deeply hierarchical Web sites.

38 A *cache*, in this context, is a local copy of data retrieved from an external source. A cache can generally be read and processed in less time than it would take to retrieve the same data from the originating source. Where data changes

(iv) the CGI script delivers the data to the Web server software, formatted for inclusion in the home page; (v) the CGI script closes its output, letting the Web server software know it is finished providing content for the home page; (vi) the Web server software embeds the formatted data in the home page; (vii) when all the CGI scripts have finished, the home page, with all embedded content, is delivered to the user's Web browser.

Meanwhile, after each CGI script finishes delivering content to the Web server, a *post hoc* subroutine checks the age of the cache file, and if it is older than a specified time period, attempts to download a new copy of the data from the Pipes feed, so as to refresh the contents of the cache file. The expiration time period for the various features during the proof of concept stage varied from one to 24 hours, depending on (i) how frequently the underlying data was expected to change; and (ii) the complexity of the Pipe feeding the data to the CGI script; and (iii) the volume of data required from Meetup.com in providing the feature. More complex, data-intensive features generally had caches that were retained for longer durations.

Each CGI script drew on a Pipe that would in turn fetch and process the data used by the CGI script. Yahoo! Pipes also employs a certain degree of data caching<sup>39</sup>. Finally, several CGI scripts were employed to provide utility functions to these Pipes that were not otherwise readily available.

## **Phase 2: Prototype**

On Wednesday, June 11, 2008, a redesigned version of the Meetup.com Web site was released. Two things occurred at this time: (i) several features of the nassauwalks.org Web site failed; (ii) the Meetup API became available. Work immediately commenced to retool the nassauwalks.org Web site: first, to restore the functionality of most of the broken features; second, once features were restored, to reimplement each feature using the Meetup API.

The listing-style calendar feature, previously drawing on the RSS calendars of the Nassau County Walking Meetup Group and B O A R D W A L K, was expanded to include walking and hiking-related Meetup meetings (but not notes<sup>40</sup>) within a 19-mile radius of the effective center<sup>41</sup> of the

---

infrequently, caches in Web applications have the additional advantage of limiting the number of requests made to the remote Web server (and thus the load placed on the server by the application). Thus caches are both expeditious and courteous.

39 The duration of caching by Yahoo! Pipes is unclear. Ideally, Yahoo! Pipes would provide a predefined module to allow developers to implement a flushable cache, of programmable duration, as part of their applications.

40 Rather than relying on the manual entry of non-Meetup events as notes on the calendar of the Nassau County Walking Meetup Group, several Yahoo! Pipes feeds were developed during the prototype phase to incorporate non-Meetup events automatically into the list-style calendar feature.

41 ZIP Codes, as such, are not geographic regions of physical space with fixed boundaries, but rather groups of addresses or postal service areas, often not renderable to a geometric polygon on a map, and subject to change as the needs of the United States Postal Service change. What this means is that a given ZIP Code does not necessarily have a “center” in any meaningful sense of that term. Different vendors use different systems for relating ZIP Codes to geographic entities, which geographic entities may or may not correlate in some meaningful sense to the physical addresses covered by a ZIP Code, and for associating a geographic “center” with said mappings. One such system is the ZIP Code® Tabulation Areas (ZCTAs™) used by the United States Census Bureau. And yes, despite the ease with which we tend to write “zip code” as if it were a generic term, ZIP Code is actually a registered trademark of the United States Postal Service. The acronym, ZIP, by the way, refers to Zone Improvement Program, first implemented by the United States Postal Service in 1963.

Westbury, New York postal ZIP Code<sup>42</sup>. In order to assemble all of the data used by the listing-style calendar and event-map features, both the JSON<sup>43</sup> and RSS variants of the `events` method<sup>44</sup> of the Meetup API were drawn on, together with the RSS calendar feed<sup>45</sup> of the Meetup groups holding said events, and the public Web pages of the same events<sup>46</sup>.

The CGI scripts were refined slightly during this phase, and several additional utility CGIs were employed.

### **Phase 3: Production**

Having successfully converted the features of `nassauwalks.org` over to the Meetup API (except where the Meetup API does not yet fully support the data requirements of said features), work then began on rationalizing the Yahoo! Pipes components serving each feature. Throughout the month of July, each Pipe was streamlined and refactored. Large, unwieldy<sup>47</sup> Pipes were reimplemented as sequences of smaller Pipes served by subpipes providing specific utility functions. The majority of these subpipes were also published to enable other developers to clone these components for their own applications.

Notably, in light of limitations of the `events` method, the implementation of the calendar-list feature was significantly reworked and streamlined.

---

42 The Westbury ZIP Code represents the approximate center of Nassau County. A radius of 19 miles captures the better portion of Queens and western Suffolk County, regions in which many of the members of the Nassau County Walking Meetup Group reside, without taking in the large numbers of walks and hikes in Manhattan and Westchester County, which fall outside the focus of this local walking Meetup group.

43 JavaScript Object Notation. A structured data format for representing simple data structures and arrays of data, JSON is rapidly becoming a *de facto* standard for exchange of data between Web 2.0 applications.

44 The Meetup API provides a number of *methods*, subroutines related to classes of data on the site (*e.g.*, events, groups, members). Each method has several variants, reflecting the format of the data returned by the method. These variants do not necessarily return the same data elements. In this example, the `events.json` method delivers numerous properties absent from the `events.rss` method, while the `events.rss` method includes a richly formatted event description absent from the more light-weight JSON version of the same method.

45 The API methods do not presently identify whether an event is a meeting or a note, and, until recently, there was no means of guessing from the Meetup API if an event was private. The RSS calendar, however, marks notes for public groups with a modified event title, of the form “(note: *title*)”. On the other hand, if the Meetup group holding an event is private, the RSS feed for the group does not return any data—which can then be interpreted to indicate a private group.

46 The Meetup API `events.json` method does not presently return the geographic coordinates of events (used here to determine if the event is within the radius for inclusion on the calendar, and so as to mark the event on the map feature). Rather, the method returns, for public events, the coordinates of the Meetup group holding said event. (Strangely, some events include, in addition to “lat” and “lon” attributes, a second pairing of “venue\_lat” and “venue\_lon” attributes. However, as these attributes are not present in the records of known public events—particularly the events of the Nassau County Walking Meetup Group, their intended functionality has yet to be determined.) This reflects an issue that predates the release of the Meetup API. The Web pages of events, before and after the May 2008 redesign, include a meta tag entitled “geo.position”, which likewise lists the coordinates of the group, rather than the event. For this reason, the proof of concept and prototype phases both parsed the Meetup event page to obtain the ZIP Code of the event, if public, and used this information to approximate the geographic coordinates of the event. See the section of this white paper concerning the `[getMeetupEventGeo]` Pipe (p. 29) for further discussion of issues related to identifying the geographic location of Meetup group events.

47 As a rule of thumb, Pipes over half a dozen modules in length should be used sparingly, and those over a dozen components should definitely be refactored. As a Pipe grows larger, the time it takes to render the Pipe on the canvas of the Pipes Editor grows excessive, and timeouts and other failures occur more frequently.

The CGI scripts drawing on these Pipes were also refactored, and the *post hoc* caching functionality revised for greater responsiveness and reliability. Each of the CGIs providing utility functions were replaced with utility subpipes that provide the same functionality from within the Yahoo! Pipes platform or from other data sources.

### **Phase 4: Pending**

Some of the data used by features of nassauwalks.org are not obtainable, or not easily obtainable, given the Meetup API method calls currently available—and it is hoped that these issues will be resolved in future. Several subpipes drawing on non-Meetup data likewise would be better implemented using public APIs, or else using new Yahoo! Pipes modules (suggested herein). The current SSI/CGI architecture will meanwhile be converted to a caching-enabled AJAX<sup>48</sup> implementation.

Additional data sources are due to be incorporated<sup>49</sup>, and the scope of nassauwalks.org is to be expanded to more fully represent both western and eastern Suffolk, as well as Queens. A calendar/newsletter e-mail feature, personalized to each member's geographic location, is also to be implemented, as well as a synchronization feature for cross-listing Meetup events (and updating those listings as and when the event listings change) via other calendaring platform<sup>50</sup> APIs.

Presently, all methods currently implemented for the Meetup API are for reading data. Write methods are promised in the future. Once these write methods are available, the following additional features are planned: (i) a process for maintaining a subset (so as to limit clutter, while still keeping members informed of events that may be of interest to them) of upcoming non-Meetup events as notes on the group calendar; (ii) a wizard to enable assistant organizers to quickly convert non-Meetup event notes to meetings; (iii) an annotation feature to allow members to leave comments about non-Meetup event notes and advise their fellow members if they plan to attend said events.

---

48 Asynchronous Javascript And XML. *AJAX* is a collection of Web technologies used to build Web applications that are more responsive and less bandwidth intensive.

49 This will involve not only building new Yahoo! Pipes feeds to draw data from existing sources, but also working with organizations that hold events to make their data available more readily available to developers, and providing a service for submitting and tracking one-off events where the provision by hosting organizations of data feeds would be impractical.

50 Presently, Meetup.com provides a wizard (an application that guides the user through the process of using specific functionality) for manually posting events to Craigslist. This tool does have certain limitations, however. First, the wizard is invoked by passing event parameters, including the entire description of the event, to the wizard using a GET query (*i.e.*, appending the encoded parameters to the URL of the wizard). Some popular Web browsers (including Internet Explorer) truncate URLs beyond a fixed length, such that the wizard receives an incomplete parameter set, and thus fails. (This issue would be resolved by passing the parameters to the wizard using the POST method. Better yet, the availability of the Meetup API means the Craigslist wizard could be reimplemented to accept the event ID as its only parameter, and then obtain on its own via the event details by way of an API method call.) Second, the wizard posts events occurring in Long Island to longisland.craigslist.org, a separate site from newyork.craigslist.org. Oddly, longisland.craigslist.org maintains a different database of events than were listed in the Long Island (lgi) section of newyork.craigslist.org—such that an individual calling up craigslist.org on their Web browser, and from there clicking on “newyork”, followed by “lgi”, would never be shown events posted via the Craigslist wizard on Meetup.com. Even more peculiar, the domain li.craigslist.org redirects to the “lgi” section of newyork.craigslist.org, again bypassing the separate longisland.craigslist.org database to which the Meetup.com wizard posts Long Island events.

## Notation Used Herein

For each feature discussed herein, we will use a simple pipeline notation<sup>51</sup>, as follows:

*source* | *component* | . . . | *component* | *feed*

## Components

Each component is rendered using the following syntax:

*(component* + . . . + *component)*

An aggregation. Used in source position to indicate that two or more sources are merged (typically by a Union module<sup>52</sup>) by the component to the immediate right.

-

Used in source position to indicate that the component to its right originates, rather than fetches, data.

*component* (*function*) . . . (*function*)

A component that invokes one or more other components as utility functions (usually, via a Loop module<sup>53</sup>).

*pipeName*

A Pipe<sup>54</sup>. This naming convention has been selected to indicate a Pipe that fetches data directly or invokes a pipeline that ultimately fetches data.

*Pipe Name*

A Pipe. This naming convention has been selected to indicate a Pipe that is conceptually similar to existing Yahoo! Pipes modules or that modifies data in some way or where the title of the Pipe is presented by a Pipe badge<sup>55</sup>.

---

51 Those familiar with UNIX shell programming will find this notation familiar. Unlike in the UNIX environment, however, a “pipe” here is not a *virtual pipe* (represented by the vertical bar), linking the standard output of the program denoted to its left with the standard input of the program denoted to its right, with execution beginning with the left-most component, and data being pushed from left to right. Rather, here, the vertical bar delimits a sequence of individual Pipes (and other components); rather than reading input and writing output blindly to virtual pipes, each component explicitly invokes the component to its immediate left, such that execution begins at the right-most component, with data being pulled, rather than pushed, to the right.

52 In Yahoo! Pipes, a *Union* module, as its name suggests, joins two or more (up to five) data feeds. The union operation appends the second feed to the end of the first feed, and so forth, and returns the results as a single feed.

53 A *Loop* module iterates over the items in a feed (or over the items in an array within the items of a feed), invoking a module or subpipe that is parameterized and/or modifies elements of the data in each such item.

54 The majority of the Pipes discussed herein have been published and made available for examination and cloning as “hosted Pipes”, following the convention <http://pipes.yahoo.com/beadsland/pipename>, without spaces or capitals. In most cases, a Meetup API key will be required to use these hosted Pipes. See [http://www.meetup.com/meetup\\_api](http://www.meetup.com/meetup_api)

55 In addition to making Pipe feeds available in multiple data formats, and allowing users to view and play with hosted Pipes, one can also embed the output of a Pipe directly in a Web page as a Pipe badge. Three types of badges are available: lists, maps, and images. The maps badge type, which is available to Pipes that include geographic coordinates in their output, is discussed below.

`pipe name`

A Pipe. This naming convention has been selected to indicate feed Pipes (see below) drawn on by a CGI script.

`[pipename]`

A feed Pipe. In feed position, a Pipe that serves as either a source or a utility function to another pipeline. In source or utility function position, a Pipe in the feed position of a pipeline previously specified herein. Pipes using any of the naming conventions above may operate as feed Pipes.

`<domain.com>`

In feed position, a Web site that incorporates the results of a pipeline. In source position, an API service or Web site from which data is fetched. In component position, a Web service by way of which the component to the right fetches the component to the left.

`method.json`

A Meetup API method, returning JSON data. See [http://www.meetup.com/meetup\\_api/](http://www.meetup.com/meetup_api/)

`script.cgi`

A CGI script. May appear in component position or utility function position.

SSI

An SSI directive. Will always appear in component position the left of the Web page that includes it.

## Implementation

What follows is a discussion of the functionality and technical implementation of each of the features of [nassauwalks.org](http://nassauwalks.org).

### **Member Count**

`<api.meetup.com> | pageMeetupJsonIterator (<pipes.yahoo.com>)`

`| [pageMeetupAPI]`

`[pageMeetupAPI] | [getMeetupGroup]`

`[getMeetupGroup] | [walk member count]`

`[walk member count] | <tinyurl.com> | mcount.cgi | SSI`

`| <nassauwalks.org>`

An admission, up front: the pipeline implementing this feature is unnecessarily complex. The entire operation of getting the member count for a group, and returning it in a string, for display at the top of the `nassauwalks.org` home page, could easily be completed with a single Pipe.

We use a sequence of Pipes here for methodological purposes—we intentionally adhere to principles of abstraction: first, to streamline maintenance by ensuring that later implementation changes propagate consistently across all features; second, to demonstrate a way of thinking about Pipes implementations that will become more relevant as we explore more complex pipes below.

### [pageMeetupAPI]

This pipeline is the core of all our features. While perhaps excessive for a feature requiring only a single record of data, its utility should be readily apparent. The `pageMeetupAPI` Pipe accepts as a parameter the query-encoded URL of an API method (in this case, `groups.json`), which it then passes to `pageMeetupJsonIterator`.

In turn, `pageMeetupJsonIterator` fetches the results of that API query. If less than 200 records are returned, the Pipe outputs said results. On the other hand, if exactly 200 records<sup>56</sup> are returned, the `page` parameter of the query-encoded URL is incremented, and our `pageMeetupJsonIterator` Pipe is invoked recursively<sup>57</sup>, using the new query-encoded URL. The results of the recursive call are then joined with the results of the initial API query, and output. A “max offset” parameter guards against time-outs in the event of excessively large result sets.

The feed returned by [pageMeetupAPI] consists of the aggregate of each iterated page of data.

---

56 This represents the maximum number of records that are returned by any Meetup API method, termed a *page*. When more than 200 records are required (and, indeed, exist for the query in question), subsequent calls to the method are made, with an additional “page” parameter. Thus, to obtain records 201-400, the same method would be called with the parameter “page=1”. Of course, there is no mechanism for read-locking the Meetup database between queries for paged data. Assuming that no database cursor is being maintained between method queries, this can result in corruption of the data obtained. If a record is added between calls, such that it would be inserted into a prior requested page, subsequent pages will be offset by one record, potentially introducing a duplicate entry (*e.g.*, as what was record 200 becomes record 201). Likewise, if a record is deleted that would have occurred on a prior requested page, subsequent requests will be negatively offset, such that data is missing (*e.g.*, as what was record 201 becomes record 200).

57 A process is *recursive* when one of the steps in said process is to repeat the process itself. “Lather, rinse, repeat” is a recursive process familiar to anyone who has read a shampoo bottle. In computer programming, recursive functions are commonly used to create simple, finite processes where a loop function proves impractical or inelegant. (“Lather, rinse, then lather again, then rinse again”) Note that, in Yahoo! Pipes, recursion is performed by calling the `pipe.run` method of the `<pipes.yahoo.com>` service with the unique ID of the Pipe, parameters, and desired output format (*e.g.*, JSON). Attempting to include a Pipe as a subpipe to itself directly may work while using the Pipes Editor, but does not when the Pipe is run outside the editor.

**[getMeetupGroup]**

The Pipe [getMeetupGroup] accepts a Meetup group topic, group number<sup>58</sup>, and Meetup API key<sup>59</sup> as parameters, and assembles a query-encoded URL of the `groups.json` method, which it then passes to [pageMeetupAPI]. The results of that subpipe are then output by [getMeetupGroup].

**[walk member count]**

Our feature feed calls upon the getMeetupGroup Pipe, with the group topic, group number tuple<sup>60</sup> (walkers, 296), and the Meetup API key. The member count attribute of the resulting group record is then included in a string, in the form “*nnn* members at last count!”, and output.

**<nassauwalks.org>**

Each of the features (except for the feature implementing a map-style Pipe badge, discussed below) follow the same pipeline structure beyond the Yahoo! Pipes components. An SSI directive in the `<nassauwalks.org>` home page executes a CGI script on the Nassau Walks server, which in turn reads a cached copy of the feed output, and applies appropriate formatting, outputting the result for inclusion in the home page.

In this case, no formatting is actually applied, and the “*nnn* members at last count!” string is simply passed back to the Web server software, which embeds same on the top of the `nassauwalks.org` home page.

---

58 Groups may be identified one of two ways in Meetup API calls: either by a large numerical ID, or by a pairing of the primary category of the group, and the group's number within that category. For instance, the Nassau County Walking Meetup Group has the ID of 448895. The same group is also identified by the tuple (walkers, 296). The later means of identification, in addition to perhaps being easier to remember, also happens to be found in the URL of the home page of the group on Meetup.com (*i.e.*, <http://walkers.meetup.com/296/>).

59 All Meetup API calls require that the application making the call provide an API key assigned to a specific API developer/Meetup member.

60 As in mathematics, where a *tuple* refers to an ordered list of values, in database theory and programming generally, a *tuple* refers to a list composed of specific components. Here, the group topic, group number tuple specifies what is known as a *primary key*, which uniquely identifies a row in a relational database table. In this case, the relational database is accessed through the Meetup API, and the table is the list of all Meetup groups. Thus, our tuple uniquely identifies a record in the “groups table” of the Meetup database. We use the term tuple, rather than primary key, in this document for two reasons: (i) in so far as we are passing parameters to Pipes and API methods, and not directly accessing a relational database, tuple more accurately describes the data structure at hand; and (ii) the word “tuple” is just more fun to say.

The *post hoc* caching routine then checks the age of the cached copy of the feed output, and if it has expired<sup>61</sup>, attempts to fetch the URL of the RSS formatted output of a feed Pipe, by way of the <tinyurl.com> service<sup>62</sup>, updating the cache file if the Pipe returns with valid output<sup>63</sup>.

## **Nassau's Largest**

```
<quickfacts.census.gov> | [getCountyByZip]

[pageMeetupAPI] | getMeetupLargestLocalGroups
  | [walk nassau meetup groups] ([getCountyByZip])

[walk member count] | <tinyurl.com> | topNassau.cgi | SSI
  | <nassauwalks.org>
```

This feature performs a simple API method call, filters the records returned, and outputs a stripped-down set of the filtered results. The top ten items in this result set are displayed in an enumerated list on the upper right-hand side of the nassauwalks.org home page.

### **[getCountyByZip]**

This utility function Pipe accepts a postal ZIP Code as a parameter, and assembles a query to the “Lookup county by place name” data tool at <quickfacts.census.gov><sup>64</sup>. The result is tokenized into individual attributes and output.

### **[walk nassau meetup groups]**

The Pipe `getMeetupLargestLocalGroups` accepts a ZIP Code, radius, max offset, and Meetup API key as parameters, and assembles a query-encoded URL for the `groups.json` method, sorting by group size, which it then passes on to `[pageMeetupAPI]`, the results are then output.

- 
- 61 Presently, except where otherwise noted, feature caches are set to expire after 24 hours, plus some random fraction of an hour (so as to encourage cache expirations to drift apart over time). Note that as this routine is *post hoc*, a cache may be retained well beyond its expiration period, only being refreshed *after* it is displayed.
- 62 Yahoo! Pipes provides a mechanism for naming hosted Pipes with human-readable URLs. No such facility appears to be available for obtaining a Pipe feed in any of various formats, where only a large ungainly parameterized URL, including the unique Pipe ID is provided. To ease development, each feed Pipe invoked by a CGI script has been submitted to the tinyurl.com service, to obtain a short, easy-to-work-with URL, which is then included as a configuration variable of said CGI script.
- 63 Pipes occasionally time out without returning data. Similarly, the Meetup API service is sometimes unavailable, in which case a pipeline will execute only to output no results.
- 64 This service is essentially a search engine against a public database, the output of which is a Web page. Although it is not anticipated that this service will become unavailable (provided, as it is, by the U.S. Census Bureau, rather than a commercial entity), it would be preferable to find a public API that provides the same service. Alternatively, while the Location Builder module within Yahoo! Pipes does parse zip codes, it does not currently return a county attribute. Adding this datum would likely be a trivial extension in functionality.

Here, the `getMeetupLargestLocalGroups` Pipe is invoked with the ZIP Code of Westbury, New York, and a radius of 19 miles. For each group returned, said group's own ZIP Code is passed to the `[getCountyByZip]` utility function. Groups in Nassau County, New York are retained, all others dropped.

Unfortunately, it seems that some group descriptions include non-RSS compliant characters, and Yahoo! Pipes was not encoding these characters when rendering our feed output as RSS. As a result, `topNassau.cgi` would fail when attempting to parse the non-complaint RSS feed. As the description is not actually required by this feature, it is replaced with an empty string.

## **Member Places**

```
[pageMeetupAPI] | getMeetupGroupMembers | [walk member places]
```

```
[walk member places] | <tinyurl.com> | topPlaces.cgi | SSI
| <nassauwalks.org>
```

This feature retrieves the member records for the group, identifies the place name<sup>65</sup> associated with each member's location, and tallies those place names. The top ten items in this result set are displayed in an enumerated list on the middle (below the fold<sup>66</sup>) right-hand side of the `nassauwalks.org` home page.

### **\*\*\* Validity of Member Place Data \*\*\***

It has recently been discovered that the validity of this data is questionable. When a prospective member of a given Meetup group, who has not previously joined Meetup.com, joins said group, they are prompted for name, e-mail, and password. The member's profile defaults to the ZIP Code of the group first joined, which most likely is not their own ZIP Code.

Thus, new members joining Meetup.com for the first time, if by way of the Nassau County Walking Meetup Group, will have Westbury (11590) shown as the place name in their profile, while those who first joined Meetup.com to participate in any of the thousands of Meetup groups in New York City, will be identified as living in New York City. That such a new member actually resides in any of the hundreds of villages and hamlets of Nassau or Suffolk counties, or in a neighborhood of Queens, will not be evident—at least until they discover that they have a profile that shows this information, and that they can change it.

---

65 Here, the name associated with the post office of the member's ZIP Code. On Long Island, this will typically be an incorporated village or city or an unincorporated community (either a hamlet of the enclosing town, or a neighborhood within a village). Long Island is rich with incorporated villages and unincorporated hamlets, together with a few cities. In Queens, unlike other boroughs of New York City, many post offices are likewise named for the neighborhood they serve.

66 As with a newspaper, where content on the bottom half of a page is typically hidden until the reader picks up and unfolds (or, in the very least, flips over) the paper, so the portion of the page that is below the bottom of the Web browser window (when the page first loads) is said to be “below the fold” or “below the scroll”, insofar as the reader must do something (namely, scroll down) before the content will be visible.

As the ZIP Code associated with a given member's profile affects what information they receive from Meetup.com, including weekly calendars of events near them and alerts of new Meetup groups they might be interested in, having incorrect information here can limit the value obtained from these features. However, not all members are so curious as to examine their own profiles, or confident in their own ability to use the Web site to update said information.

From the perspective of group organizer, where the mission of the group includes encouraging people to walk locally in their own communities, having accurate data regarding where large numbers of members live is essential, as it guides initiatives to seed and foster the development of new locally-focused walking Meetup groups.

In light of the foregoing, one of the features pending development is a personalized e-mail calendar/newsletter of local walking events, drawing off the existing Walking Calendar feature (discussed below). The e-mail newsletter will provide each member with a list of events near them, based on the ZIP Code in their respective profile, and will advise members of what their profile claims their ZIP Code is, encouraging them to update this information if it is incorrect.

### **[walk member places]**

The Pipe `getMeetupGroupMembers` accepts group topic, group number, max offset, Meetup API key, and an optional list of member IDs as parameters, and assembles a query-encoded URL for the `members.json` method, which it then passes on to `[pageMeetupAPI]`, the results are then output.

Here, the `getMeetupGroupMembers` Pipe is invoked with the tuple `(walkers, 296)`. For each member record returned, the place name associated with said member is elevated to the title attribute. Unique place names are tallied, and the description of each resulting place name record is set to the repeat count for said place name. As the resulting records no longer refer to specific members, the link attribute is cleared. These items are then output, sorted by repeat count.

### ***Largest Walking***

```
[pageMeetupAPI] | getMeetupTopicGroups | getMeetupStateTopicGroups
  | [walk groups]

[walk groups] | <tinyurl.com> | topWalking.cgi | SSI
  | <nassauwalks.org>
```

This feature retrieves the walking groups in New York State. The top ten items in this result set are displayed in an enumerated list below the middle right-hand side of the `nassauwalks.org` home page.

**[walk groups]**

The Pipe `getMeetupTopicGroups` accepts `group topic`, `max offset`, and `Meetup API key` as parameters, and assembles a query-encoded URL for the `groups.json` method, with the results to be ordered by member count, which it then passes on to `[pageMeetupAPI]`, the results are then output.

Likewise, the `getMeetupStateTopicGroups` Pipe accepts `group topic`, `max offset`, and `Meetup API key`, as well as a state code, in the form of the two letter United States Postal Service abbreviation for a U.S. state<sup>67</sup>. The Pipe `getMeetupTopicGroups` is invoked, and for each group record returned, the ZIP Code of the group is passed to a `Location Builder` module<sup>68</sup>. The state of the group's ZIP Code is compared to our state code, and discarded if not. The remaining group records are output.

Here, the `getMeetupStateTopicGroups` Pipe is invoked with the topic “walkers”<sup>69</sup> and state code “NY”, and the results are output, sorted by member count.

**Album Thumbnails**

```
[pageMeetupAPI] | getMeetupGroupPhotos | [walk photos]
```

```
[walk photos] | <tinyurl.com> | photo.cgi | SSI | <nassauwalks.org>
```

This feature retrieves the album thumbnails of the Nassau County Walking Meetup Group, which are displayed on the left-hand side of the `nassauwalks.org` home page.

**[walk photos]**

The Pipe `getMeetupGroupPhotos` accepts `group topic`, `group number`, `max offset`, and `Meetup API key` as parameters, and assembles a query-encoded URL for the `photos.json` method, which it then passes on to `[pageMeetupAPI]`, the results are then output. By default, the `photos.json` method returns sorted reverse chronologically by album creation date.

Here, the `getMeetupGroupPhotos` Pipe is invoked with the tuple `(walkers, 296)`. For each resulting record, a description string is assembled consisting of the album thumbnail and the album title

---

67 Pennsylvania, Massachusetts, Virginia, and Kentucky, technically, are commonwealths, not states, but the two letter abbreviations work for those jurisdictions nonetheless. Although untested, we assume the USPS abbreviations for the District of Columbia and the eight territorial possessions of the United States also work.

68 In Yahoo! Pipes, a *Location Builder* module attempts to determine the geographic location identified by a string of text. Although designed to parse street addresses, this module will also recognize ZIP codes without any other identifying information, returning the state abbreviation of the ZIP Code, as well as the latitude and longitude approximating the center of the ZIP Code area.

69 For reasons discussed in Footnote 113, this query only returns groups with “walkers” as the primary topic. Subsequently, an RSS feed of walking Meetup groups, which includes groups with “walkers” as an alternative topic, has been found. This feature will eventually be updated to use said RSS feed.

(typically, the name of the Meetup group event<sup>70</sup> at which the photos were taken), linking to the album page. These records are then output.

## Walking Quotes

```
<www.meetup.com> | [getMeetupNewBios]

[pageMeetupAPI] | [getMeetupGroupMemberReviews ]

<topic.meetup.com> | [getMeetupGroupMemberPhoto]

- | [Quote Builder]

([getMeetupNewBios] + [getMeetupGroupMemberReviews])
  | [walk quotes] ([getMeetupGroupMemberPhoto]) ([Quote Builder])
  | <tinyurl.com>71

(<tinyurl.com> + <tinyurl.com> + <tinyurl.com> + <tinyurl.com>72
  + quotes.txt73 + inline74) | quote.cgi

quote.cgi | SSI | <nassauwalks.org>
```

This feature aggregates three sources of data: (i) the bios<sup>75</sup> of new members of the Nassau County Walking Meetup Group; (ii) the review comments of Nassau County Walking Meetup Groups where members rated a Meetup group event highly; and (iii) walking-related quotes from famous persons and historical figures. One of these items is then randomly displayed on the nassauwalks.org home page. If the item displayed is from the first or second source above, and the member quoted as a profile photo, their photo is shown next to their respective bio or review comment.

With each subsequent load of nassauwalks.org, a random quote is selected from the resulting aggregated feed, and displayed in near the center top (just below the Where We're Walking feature) of the main content area of the nassauwalks.org home page.

<sup>70</sup> Technically, Meetup.com, as such, does not hold any events. Rather, Meetup.com hosts Meetup groups, which in turn, hold events.

<sup>71</sup> We've had to break from notational convention here, to be explained below. Apologies for any confusion.

<sup>72</sup> Uh, we don't actually call this service four times, but we do use the results more than once. See below.

<sup>73</sup> Oops. Still another break from convention!

<sup>74</sup> Yes, we are very misbehaved. All will be explained...

<sup>75</sup> When any Meetup.com member joins a Meetup group, they may specify optional information to display to fellow members of that group, including a profile photo and *bio*, or summary self introduction. Organizers can also create questions that members may optionally answer, their answers becoming part of their group profile.

**[getMeetupNewBios]**

The Pipe [getMeetupNewBios] accepts group topic and group number as parameters, and assembles the URL of the What's New RSS feed for a Meetup group<sup>76</sup>, which it then fetches. Of the RSS results received, all but the items for new members are discarded. Where the member has a bio, the prefix "BIO" is removed, otherwise, the item description is set to indicate that no bio was provided. The "Member:" prefix is likewise removed from the item title, and copied to the member name attribute. All results are then output.

**[getMeetupGroupMemberReviews]**

The Pipe [getMeetupGroupMemberReviews] accepts group topic, group number, max offset, event rating, max offset, and Meetup API key as parameters, and assembles a query-encoded URL for the `comments.json` method, restricting output to only those ratings that have review text<sup>77</sup>, which it then passes on to [pageMeetupAPI]. For each item, the comment attribute is copied to the item description, the title attribute to the member name. All results are then output.

**[getMeetupGroupMemberPhoto]**

The Pipe [getMeetupGroupMemberPhoto] accepts the URL of a group member's profile page, and fetches same, extracting the URL of the member's group photo, if any<sup>78</sup>, which is then output.

**[Quote Builder]**

The Pipe [Quote Builder] accepts a header string, optional photo URL, quotation string, and member name, which parameters are combined into a formatted HTML snippet, and output.

**[walk quotes] | <tinyurl.com>**

The Pipe [walk quotes]: (i) invokes [getMeetupNewBios] with the tuple (walkers, 296), (ii) invokes [getMeetupGroupMemberReviews] with the tuple (walkers, 296), a rating of 5, and Meetup API key; (iii) assigns a header string appropriate header string attribute to each of the items returned by both feed pipes; and (iv) aggregates the results into a single feed.

For each item: (i) the [getMeetupGroupMemberPhoto] Pipe is invoked with the link to the given member's profile page; (ii) the [Quote Builder] utility function Pipe is then invoked with header string, group member photo (if any), item description (*i.e.*, the member's review comment or bio); and (iii) result from [Quote Builder] is set as the item description. The resulting items are then output.

---

<sup>76</sup> Of course, much the same information could be obtained using the `members.json` method—the result set could be filtered to only include items occurring in a certain time period, or truncated after an arbitrary number of reverse chronologically sorted items. We use the RSS feed mostly because it is there.

<sup>77</sup> Despite its name, the `comments` method actually returns member ratings, which may or may not have associated comments, unless a "review\_only" flag parameter is explicitly set.

<sup>78</sup> This Pipe would be obviated if the member's group photo were included as an attribute of the `members.json` method, and, optionally, the `comments.json` method.

Our break from convention here—by placing a <tinyurl.com> component in feed position—is to simplify our description of the `quote.cgi` pipeline, below.

### **quote.cgi**

The script `quote.cgi` retrieves the [walk quotes] feed, by way of a short-form URL provided by the <tinyurl.com> service. An array is constructed, and each item from the [walk quotes] feed is copied to that array four times.<sup>79</sup> Also added to this array is a database of quotations about walking by various famous persons and historical figures<sup>80</sup>, each attributed to the Web site where it was first found. Finally, a few heavily formatted quotations that do not fit easily in our quotes database are added directly from inline<sup>81</sup> strings within the `quote.cgi` script.

An item is then randomly selected from our array and returned.

### **Local Calendar**

```
- | [parseMeetupEventUrl]

<www.meetup.com> | getMeetupEventCalendarItem ([parseMeetupEventUrl])
    | [getMeetupEventType]

<topic.meetup.com> | [getMeetupEventGeo]

[pageMeetupAPI] | getMeetupGroupEvents | [getRichMeetupGroupEvents]
    ([getMeetupGroup]) ([getMeetupEventType]) ([getMeetupEventGeo])

<?> | [callDistanceJsonUrl]

- | [Proper Case] (<pipes.yahoo.com>)
```

---

<sup>79</sup> Presently our database of walking-related quotations by famous persons and historical figures has significantly more records than all member review comments and new member bios combined. Mixing member quotes with historical quotes at a ratio of four parts to one ensures that member quotes appear often enough to have made the whole exercise of aggregating this data worthwhile.

<sup>80</sup> Our quotes database can be viewed at <http://www.nassauwalks.org/quotes.txt>.

<sup>81</sup> That is, embedded directly into the Perl script, in this case, using the <<EOF string notation.

```

[pageMeetupAPI] | getMeetupLocalTopicGroups
    | getRichMeetupLocalGroupEvents ([parseMeetupEventUrl])
        ([getRichMeetupGroupEvents])
    | getRichMeetupLocalMeetings ([callDistanceJsonUrl])
    | [Nassau County Walking Meetup Events] ([Proper Case])

- | [recurMonthlyWalks]

- | [getMonthDayOfWeeks]

[getMonthDayOfWeeks] | [getMonthOrdinalDayOfWeek]

- | testInDateRange | [testInFloatingDateRange]

- | [floatToUSAwDST] ([getMonthOrdinalDayOfWeek])
    ([testInFloatingDateRange])

- | [recurWeeklyWalks] ([getMonthDayOfWeeks]) ([floatToUSAwDST])

- | [oldWestburyWalks]

- | [muttontownWalks]

([recurMonthlyWalks] + [recurWeeklyWalks] + [oldWestburyWalks]
    + [muttontownWalks]) | [walks not on meetup] ([testOldDate])

<www.discoverylongisland.com> | [discoverWalks]

([Nassau County Walking Meetup Events] + [walks not on meetup]
    + [discoverWalks]) | [NassauWalks Calendar]
    ([callDistanceJsonUrl])

[NassauWalks Calendar] | [Where We're Walking (Text)]

```

[Where We're Walking (Text)] | <tinyurl.com> | rss.cgi

rss.cgi | SSI | <nassauwalks.org>

## Overview

Comprised of well over two dozen components<sup>82</sup>, this is far and beyond the most complex feature of nassauwalks.org. In order to expedite the drafting of this white paper, a number of the constituent components of this feature have been left as Phase 2 prototypes, unquestionably in need of refactoring. The naming scheme is also due for some revision.

This feature aggregates three high level sources of data: (i) data-rich records representing walking- and hiking-related Meetup events in the local area; (ii) a collection of walking-related events not found on Meetup.com; and (iii) local walking- and hiking-related events listed by the Long Island Convention & Visitors Bureau and Sports Commission<sup>83</sup>.

The first of the foregoing data sources flows from an extended pipeline that first fetches candidate Meetup group events, adds data elements that will be required in later stages, and then based on that added information, filters out items that do not meet our criteria. The second data source largely constructs from scratch a collection of (a) recurring events and (b) events on the current seasonal calendar of two popular venues. The third data source fetches and organizes search results from an online database promoting events held by a wide range of organizations on Long Island.

The resulting aggregated feed is truncated, excluding events occurring over a month in the future, and pulled down to a CGI script on the nassauwalks.org server, and then incorporated into the home page, comprising substantially all of the main body of the page, below the Where We're Walking and Walking Quotes features.

### \*\*\* Volume and Scale \*\*\*

In addition to stretching our notational framework to its limits, this feature also presents challenges for the technologies that undergird it. Timeouts and other transient errors in component Pipes lead to

---

82 In addition to the issues discussed below, the sheer number of components that go into this feature bring to light a weakness of the Yahoo! Pipes platform as currently implemented. There are no tools for organizing and managing large numbers of Pipes. Rather, a developer can view paged list of all their pipes, organized reverse chronologically by when last edited, or else a paged list of their “favorite” pipes, likewise sorted. The Pipes Editor similarly provides only for “Favorites” and “My Pipes”, and when either of these lists gets to be longer than screen real estate, can fail to display any Pipes at all (showing only a blank area where the Pipes would have been listed). A related issue is that there is no mechanism for deprecating Pipes, short of deleting them outright, or of checking to see if a Pipe in one's collection currently has dependencies among other Pipes.

83 A New York non-profit membership corporation, formed in 1978, organized to promote the Nassau-Suffolk region as a destination for visitors, meetings, conventions, trade expositions, and sporting events. The Long Island Convention Bureau and Sports Commission's Web site, in furtherance of its mission to provide tourism-related economic development, travel, and tourism services for the Nassau and Suffolk county region, maintains an online database of upcoming events, organized by region (North Shore, South Shore, Central Suffolk, North Fork, South Fork, as well as Nassau County and Suffolk County).

occasional omissions, errors, and phantoms<sup>84</sup> in our resulting data feed. Meanwhile, while we rely on the caching facility of Yahoo! Pipes to limit successive identical queries to the Meetup API service, it is possible that asynchronously invoked fetch operations may circumvent this capacity, simply by arriving in too short a span to allow the Pipes engine to recognize requests as duplicative of one another.<sup>85</sup>

The Meetup API service, likewise, seems especially sensitive to the volume of queries<sup>86</sup> involved in assembling the data used by this feature, such that—during periods of active testing and debugging, when the same Pipe may be run repeatedly in a short time frame with different parameters—the Meetup API service has often become unresponsive<sup>87</sup>, such that development efforts were stymied waiting for service to be restored.<sup>88</sup>

In light of these issues, options for interposing and an external flushable JSON caching service between pipeline components are currently being investigated, with an eye toward incorporating such buffering functions during Phase 4 pending development.

### **[parseMeetupEventUrl]**

This utility function Pipe accepts the URL of a Meetup group event as a parameter, parses same to determine the group topic, group number, and event ID, and outputs an item containing these attributes.

### **[getMeetupEventType]**

The Pipe `getMeetupEventCalendarItem` accepts the URL of a Meetup group event as a parameter, passes this to `[parseMeetupEventUrl]`, and uses the output of that utility function to construct the URL of the RSS calendar feed of the Meetup group holding said event. The RSS calendar

---

84 Several components perform tests on data attributes, filtering out item that do not meet test requirements, or obtain data to be used in such tests. If such a component (or a module within such a component) times out or otherwise transiently fails, a test may falsely preserve an item that otherwise would have been dropped. Such *phantoms* may appear in a feed temporarily, only to disappear when the cache(s) serving that feed are flushed and the pipeline components are freshly executed.

85 This is only a hypothesis, guessed at to explain other application behavior discussed in this section.

86 Analyzing data from dozens of local walking- and hiking-related groups, each of which may have any number of events on their calendars, this feature can easily exceed Meetup's default 100 requests per hour developer limit in a single run, such that development of this feature was not even attempted until arrangements were made to have these limits raised. That said, many of the queries presently made by this feature serve to compensate for incompleteness of data returned from API methods as currently implemented.

87 Otherwise valid queries passed directly to the Meetup API service from a Web browser during these outages returns one of the following messages: “Sorry, there was a problem We weren't able to give you the page you were looking for. Please check the address and try again. If the problem persists and you think it may be an issue on our end, you can email us at support@meetup.com.” or “Our server wasn't able to process your request. This may be a temporary problem due to an upgrade, so please try again in a few minutes. If the problem persists, contact us.” The same query attempted 10-20 minutes later will return the expected data output.

88 Such unresponsive periods also occur, less frequently, when development is not actively in progress. Unfortunately, if the Meetup API service is unresponsive when a Pipe component querying the service refreshes its cache, the Pipe then caches an empty result set, which propagates forward through the pipeline. The caching mechanism in the `rss.cgi` component, meanwhile, is not currently configured to recognize whether the feed retrieved contains any Meetup events, which means that the `nassauwalks.org` home page occasionally shows only non-Meetup events until such time as the Meetup API service is restored, the Pipes querying that service refresh their caches, and the `rss.cgi` component refreshes its own cache from the refreshed Pipes.

feed is fetched, and all items not matching the event ID sought are discarded. The remaining item is output. (In the case where the group holding the event is private, such that the RSS calendar of said group is not available, no items are output.)

In turn, the [getMeetupEventType] Pipe accepts the URL of a Meetup group event as a parameter, and passes this to `getMeetupEventCalendarItem`. If an item is returned, the “private” attribute is set to “false”, and the title attribute is parsed, to determine if the event is a calendar note, at which point the “note” attribute is set accordingly. The item is then output. On the other hand, if no item is returned by `getMeetupEventCalendarItem`, an item is built with a lone “private” attribute with the value “true”<sup>89</sup>, and output.

This component pipeline would be obviated were equivalent attributes provided in the records returned by the `events.json` method. Currently, there is no way, using the raw data returned by this method, to differentiate calendar notes from calendar meetings, nor identify which events are held by private groups.

### [getMeetupEventGeo]

The Pipe [getMeetupEventGeo] accepts the URL of a Meetup group event and a threshold value as parameters. It attempts to fetch the public Web page of the Meetup group event, extracting the address of the event venue, if listed. Extraneous characters are removed, and the resulting string is passed to a Location Builder module. If the module outputs with a quality<sup>90</sup> attribute greater than our threshold value, this data is output. If not<sup>91</sup>, the event ZIP Code alone is passed to a Location Builder module, and the results of that module are output.

If the event venue is not available on the Web page of the Meetup group event, or else the page is not successfully retrieved, no output is produced.

This component pipeline would be obviated were equivalent data provided in the records returned by the `events.json` method. Although records returned by the `events.json` method include “lat” and “lon” attributes, as of July, these refer to the geographic coordinates associated with the ZIP Code of the Meetup group holding the event, rather than the event venue.<sup>92</sup>

---

89 This can result in false positives. Specifically, in the case of a Meetup group meeting scheduled to begin with the last hour (it may actually be two hours). There rules for inclusion of currently occurring events are different for the `events.json` method and a group's RSS calendar, such that `events.json` will return a record for an event that has started only within the last hour, but the RSS calendar will not. As we use the absence of an RSS item for a given event to indicate the privacy status of the Meetup group holding the event, this means events that have already begun will inevitably be identified (inaccurately, in many cases) as private.

90 In addition to geographic information, such as latitude, longitude, and city name, the Location Builder module also returns a quality attribute, indicating the algorithm's confidence in the accuracy of the results output.

91 This is one of several Pipes discussed herein that take advantage of the manner in which the Union module appends feeds, one to the next, to simulate an if-then-else construct. However, unlike a traditional if-then-else construct, where the alternative case is only executed if the test for the primary case fails, our split-union-truncate construct executes both cases in parallel, and then discards the alternative case if the test for the primary case is successful. This is an imperfect simulation, as a transient timeout in one of the modules of the thread for the primary case—executing asynchronously with the alternative case thread—can result in a false “else” result, typically producing phantom data further down the pipeline.

92 For a further discussion of the current state of geographic data returned by the `events.json` method, see Footnote 46.

**\*\*\* On the Issue of Event Geodata \*\*\***

For any given Meetup group event, one of five scenarios applies: (1) the Meetup group holding the event is private; (2) the Meetup group holding the event is not private, but the event is a note; (3) the Meetup group holding the event is not private, the event is a meeting, but the venue of the event is only shown to group members; (4) the Meetup group holding the event is not private, the event is a meeting, and the venue of the event is shown to non-members, but has not yet been selected by the organizer; (5) the Meetup group holding the event is not private, the event is a meeting, the venue of the event is shown to non-members, and said venue has been selected by the organizer.

Scenario (1) is trivial (or would be, if events of private Meetup groups were identified as such<sup>93</sup>), as private groups do not share any details about their events with non-members, and thus the geographic location of said event would not be available without joining the group.<sup>94</sup>

Scenario (2) is, on its face, also trivial, or likewise would be, if whether events were notes or meetings were easily determined from the attributes returned by the `events.json` method.<sup>95</sup>

Scenario (4) would be trivial, if meeting-type events that do not have a venue selected were easily identifiable as such. (A flag to that effect in the JSON item for the event would do the trick.)

Scenario (5) would be trivial, if event location data were consistently provided in the records returned by the `events.json` method. By consistently, we mean that if a Meetup group is public, and a venue of a given event held by that group has been selected by the organizer, the geographic location of said venue is at all times identified in the JSON item for that event. Presently, this is not the case.

Scenario (3) is non-trivial. Meetup organizers often hold events for which the geographic location of the event is a key feature of why members might want to participate. This is especially the case with walking and hiking Meetup groups, where the “where” of an event plays much the same role as the subject for discussion in a philosophy group or the book title and author in a book club. Organizers of walking and hiking groups tend, therefore, to talk up the “where” of the event, both in the event title (e.g., “Walk and Picnic at Jones Beach”, “Hike at Cold Spring Harbor State Park”) and in the text description of the event.

At the same time, a Meetup organizer may choose to show the venue (that is, the exact street address) at which members are supposed to meet only to members of the Meetup group. This is typically done to encourage non-members to join the group<sup>96</sup> and RSVP for the event—rather than just show up at the

---

93 See discussion of the event type attributes supplied by the `[getMeetupEventType]`, above.

94 In our implementation, the events of a Meetup group that is private are assigned the latitude and longitude associated with the ZIP Code of said group. Additionally, a boilerplate description is assigned to the event, advising the reader that they can apply to join the private group if they are interested in learning more. To avoid near duplicate, essentially non-informational, entries on the group calendar, any private group that has more than one event on its calendar is filtered down to a single entry, the first event chronologically.

95 Again, see discussion of `[getMeetupEventType]`, above. That said, it should be noted (ahem) that the “note” feature is often used by Meetup organizers to advise their group's members of events hosted by other organizations. Although not occurring as an official meeting of the group, it is conceivable that some future release of the Meetup.com application might provide for identifying the venue (or else least the general geographic location) associated with such notes, which data would then be available to developers. Here, notes, lacking any meaningful geographic data (or a reliable way to extract such data) are simply discarded.

96 Or, if the Meetup organizer requires that prospective members be approved before they are allowed to join, that said non-members apply and be approved for membership in order to then RSVP for the event.

event unannounced. In other words, where the Meetup group is not private, the Meetup organizer openly shares details about their events with non-members, but may withhold one key piece of information (the exact building number or intersection at which the group is meeting) so as to provide an incentive for prospective members, drawn in by the event title and description, to join (or apply to join) and RSVP for the event.

That this decision, to hold out specific information as an incentive to prospective members, should lead to general information about the event (where it is located) being obfuscated from applications that would otherwise promote that event to additional prospective members, is both ironic and counterintuitive. When an organizer announces a publicly-listed event with “Jones Beach” in the title, it is reasonable to assume that they would not object to that event being listed in a calendar of events occurring in Nassau County<sup>97</sup>, with suitable links back to the event listing on Meetup.com, of course. Likewise, it is a good possibility that the organizer of an event described, again publicly, as occurring in “Cold Spring Harbor State Park”<sup>98</sup> would not object to that event being marked on a map of upcoming events as occurring on the North Shore of Long Island.

For the purposes of such applications, the exact street address at which members will be gathering when the event begins is immaterial. That a Jones Beach event is occurring in the general environs of Wantagh, that a Cold Spring Harbor event is happening in the northern part of the Town of Huntington, is information enough. Computer programs, however, aren't terribly good at extracting the semantic content of place names from arbitrary text—certainly not as good at such a task as a human being otherwise familiar with the general geography of a region.

There is a reasonable compromise, that addresses concerns of privacy, while still honoring the intent of organizers who want to share their events with prospective members, even as they hold the exact address of said events in reserve as an incentive for those prospective members to join and RSVP. The compromise is this: include the ZIP Code of the venue as an attribute of the `events.json` method.

Given that ZIP Codes represent nothing more than a group of addresses (see Footnote 41), typically spread over an area measured in square miles<sup>99</sup>, yet lacking clear geographic boundaries, and the geographic (latitude and longitude) coordinates assigned to any given ZIP Code will vary from one geographic information system to another (*i.e.*, such coordinates are essentially arbitrary)<sup>100</sup>, revealing

97 Jones Beach State Park is located in Wantagh, New York, a hamlet of the Town of North Hempstead, in Nassau County.

98 Located in the Town of Huntington, in Suffolk County, New York.

99 The exceptions being those ZIP Codes that refer to a single large building or to post office boxes within a single post office. The later case is obviously not relevant to the current discussion of venues. The former could, in extreme cases, present privacy concerns. For example, the Empire State Building in New York City has its own ZIP Code. If an event organizer holding an event at a restaurant located in the Empire State Building makes a passing reference to eating a meal in the event description, but does not show the event venue to non-members, it is conceivable that a sufficiently determined non-member, armed only with the ZIP Code of the event and a Global Positioning System (GPS) unit, could locate the event without first joining the Meetup group, simply by exercising the process of elimination. (There are a grand total of two restaurants, a sushi bar, and three coffee shops physically located in the Empire State Building.) However, so-called “single point” ZIP Codes are typically identified (or else identifiable, from other data or data omissions) in many vendors' geographic ZIP Code databases. The `events.json` method could easily filter out such instances, populating the “zip” attribute only of those events not being held at such single-point venues.

100 To make this point clearly, the solution suggested would have the `events.json` method return only the ZIP Code of the event venue, in the cases of venues shown only to members of the Meetup group holding the event (the method could still also return the latitude and longitude of the venue in those events for which the venue is actually shown to non-members). It would then be the responsibility of each developer to obtain geographic data based on the ZIP Code

the ZIP Code of a venue, not otherwise identified for non-members, would not even superficially violate the privacy of the group.

Meanwhile, those “essentially arbitrary” geographic coordinates are still “close enough” to satisfy the requirements of mashup applications that present local event listings and/or regional maps of upcoming events, thereby promoting Meetup group events to prospective new members that might not otherwise find those groups or their events. Such applications don't need a geographic location so specific as to be colloquially “in the ball park”. In the general vicinity of the village or community is more than sufficient.

### [getRichMeetupGroupEvents]

The Pipe `getMeetupGroupEvents` accepts a group topic, group number, max offset, Meetup API key, and optional event ID as parameters, and assembles a query-encoded URL for the `events.json` method, which it then passes on to `[pageMeetupAPI]`. All results are then output.

Here, the `[getRichMeetupGroupEvents]` Pipe likewise accepts a group topic, group number, max offset, Meetup API key, and optional event ID as parameters, and passes the same parameters to `getMeetupGroupEvents`. Thereafter, the following steps are performed:

1. for each item, the `[getMeetupGroup]` Pipe is invoked<sup>101</sup> with the group topic, group number, and Meetup API key, and the result added to said item as a “group” attribute<sup>102</sup>;
2. for each item, the `[getMeetupEventType]` Pipe is invoked with that item's Meetup event URL, and the result added to said item as a “type” attribute<sup>103</sup>;
3. for each item with its private attribute set to “true”, the time attribute is tested to see if the event was already scheduled to have begun, and if so, it is dropped<sup>104</sup>;
4. for each item with its private and note attributes set to “false”, the `[getMeetupEventGeo]` Pipe is invoked with the URL of the event, and the result added to said item as a “y:location”<sup>105</sup> attribute; and

---

provided, using whatever vendor, and getting whatever results, may be available for that purpose.

101 As noted above, we are relying on the caching mechanism of Yahoo! Pipes to avoid repeat calls to the Meetup API service—essentially treating the Pipes cache as runtime data storage. Our assumption that `[getMeetupGroup]` is acting here as a fetch-once, read-many-times data object only holds, however, if the caching mechanism successfully blocks additional instances of the same fetch until it has finished committing the results of the first fetch to the cache. Alternatively, if Yahoo! Pipes were to provide a mechanism whereby the first item of an arbitrary feed (or all the items in a feed, treated as an array) would be treated as an input, such that said item could be assigned as an attribute of one or more other items (much in the way the Date Builder now works, albeit as a special case), the `[getMeetupGroup]` subpipe would only need to be invoked once within any instance of our `[getRichMeetupGroupEvents]` Pipe..

102 This operation would be obviated if the attributes of the Meetup group holding each given event, including group name, group photo URL, and group geographic coordinates, were returned as additional attributes of such events, either automatically, or upon the inclusion of a parameter flag (e.g., “group\_data=1”) in the `events.json` query.

103 Again, and as discussed in the discussion of `[getMeetupEventType]`, this would be obviated if the information obtained here were available as attributes of the items returned by the `events.json` method.

104 This is done to deal with false positives in the “private” attribute. See discussion of `[getMeetupEventType]` pipe, above.

105 Yahoo! Pipes converts feeds with `y:location.lat` and `y:location.lon` to GeoRSS format, a standard for encoding location data in RSS items.

5. for each item with no `y:location` defined (either because the item is private, a note, or both, or because the `[geoMeetupEventGeo]` Pipe failed to return), the values in `group.lat` and `group.lon` are copied to `y:location.lat` and `y:location.lon`.

The results are then output.

### [callDistanceJsonUrl]

The `[callDistanceJsonUrl]` utility function Pipe accepts an event latitude, event longitude, center ZIP Code, URL of a Web service that performs a distance calculation (returning its results in JSON format), and the path to a distance value the the JSON results of said Web service. After passing the ZIP Code to a location builder module, the parameters `lat1`, `lon1`, `lat2`, `lon2` are appended to the Web service URL, and the resulting distance calculation fetched. We pass the distance through a Simple Math module<sup>106</sup>, adding 0 to it so as to recast its data type from string to number<sup>107</sup>. The result is then output.

This module is peculiar, in that it calls an Web service supplied by the developer. Distance calculations tend to be expensive<sup>108</sup>, and thus we've chosen not to expose the Web service we are using for this purpose.<sup>109</sup>

Our implementation does suggest one interesting possibility: the algorithm used to perform the distance calculation is left open. Here, we are using a great-circle distance calculation<sup>110</sup>, but there is no reason that a service that calculates travel distance could not be used.

### [Proper Case]

The `[Proper Case]` utility function Pipe accepts a string as a parameter, and passes same through a series of Regex module<sup>111</sup> rules. To work around a limitation in the Yahoo! Pipes implementation of regex backreferences<sup>112</sup>, we recursively call the `[Proper Case]` Pipe with the modified string until no further modifications can be made. The result is then output.

---

<sup>106</sup> As its name suggests, the Simple Math module in Yahoo! Pipes performs simple arithmetic operations.

<sup>107</sup> Without doing this, subsequent Filter module operations will fail, as the “is greater than” and “is less than” operators only work if Yahoo! Pipes recognizes the attribute being compared to be a number.

<sup>108</sup> Both computationally and economically. Several Web services are available that will perform a fixed number of distance calculations per time period, or else greater numbers of calculations for a subscription fee.

<sup>109</sup> Yahoo! Pipes allows developers to specify private strings. These strings operate within the context of a developer's Pipe and the subpipes invoked with the private string as a parameter, but are neither shown to others nor copied when the Pipe is cloned.

<sup>110</sup> A trigonometric function that calculates the shortest distance between two points on a perfect sphere. Of course, the Earth isn't perfectly spherical, but as it is, this function does a good job at getting an “as the crow flies” approximation of distance between two places.

<sup>111</sup> A *regex*, or regular expression, is a formal language for concisely identifying patterns in a string. The Regex module of Yahoo! Pipes provides developers with a partial implementation of this functionality.

<sup>112</sup> Backreferences in regular expressions allow for the reuse of portions of a matched string. Unfortunately, the implementation in Yahoo! Pipes does not fully support the use of backreferences with a global tag. Specifically, the Regex module uses the backreference string from the first iteration in all subsequent iterations of a global operation, rather than uniquely tying each backreference to a specific iteration.

## [Nassau County Walking Meetup Events]

The Pipe `getMeetupLocalTopicGroups` accepts a group topic, ZIP Code, radius, max offset, and Meetup API key as parameters, and assembles a query-encoded URL for the `groups.json` method, sorting by group size, which it then passes on to `[pageMeetupAPI]`, the results are then output.

The `getRichMeetupLocalGroupEvents` accepts a comma-delimited group topic list, ZIP Code, radius, max offset, comma-delimited group URL list, and Meetup API key as parameters. For each group topic, if any, the `getMeetupLocalTopicGroups` Pipe is invoked with said group topic, and the ZIP Code, radius, max offset, and Meetup API key. For each group URL, if any, an item is created with said URL as its single attribute.<sup>113</sup> These items and the results of the `getMeetupLocalTopicGroup` operations are aggregated into a single feed.

For each item: (i) duplicate items are dropped; (ii) groups with a negative group number<sup>114</sup> are dropped; (iii) the `[parseMeetupEventUrl]` utility function Pipe is invoked with the Meetup group URL<sup>115</sup>; and (iv) the `[getRichMeetupGroupEvents]` Pipe is invoked with the group topic, group number, max offset, and Meetup API key as parameters. All resulting items are sorted chronologically and output.

The Pipe `getRichMeetupLocalMeetings` accepts a comma-delimited group topic list, ZIP Code, group radius, max offset, comma-delimited group URL list, event radius, URL of a distance calculator service, path of the return value of said service, and Meetup API key as parameters. The Pipe `getRichMeetupLocalGroupEvents` is invoked, with the comma-delimited group topic list, ZIP Code, group radius, max offset, comma-delimited group URL list. Events with a note attribute of “true” are dropped.

For each of the remaining items: (i) the `[callDistanceJsonUrl]` utility function Pipe is invoked, with the URL of our distance calculator service, the JSON path for same, our ZIP Code, and the latitude and longitude of the event item; and (ii) the distance value returned is compared with the event radius parameter, such that those events outside our event radius are dropped.<sup>116</sup> The remaining items are sorted chronologically and output.

---

<sup>113</sup> This explicit inclusion of specific group URLs was included to compensate for a functional limitation of the `groups.json` method, which returns items only for those groups where the topic sought is the primary topic. We allow for the entry of explicit Meetup group URLs so that known groups with “hiking” and/or “walkers” only as alternative topics, which will not be returned by queries seeking those topics, might still be included for consideration in our feed. Since this pipeline was engineered, topic-specific RSS feeds have been found on the Meetup site (these feeds were not apparent immediately after the relaunch), which include all groups that match either a primary or alternative topic.

<sup>114</sup> Early on in Phase 2 development, it was discovered that some items referred to non-existent groups, including a mysterious “Carle Place Walking Meetup”, with a large negative integer in its group URL.

<sup>115</sup> We are taking a shortcut here. Although not an “event” URL, our subpipe still successfully extracts group topic and group number.

<sup>116</sup> This entire pipeline is constructed to overcome a limitation of the `events.json` method. As discussed elsewhere in this white paper, this method currently returns, for the events of non-private groups, the latitude and longitude of the group, not of the event venue. When called with the “radius” parameter, the `events.json` method likewise returns the events of groups falling within the given radius, rather than the events whose venues fall within said radius. Thus, we have implemented a pipeline that cast a wide net (all the groups within a specified group radius) to find groups that potentially could hold events near enough to be included, get all events being held by said groups, and then determine, as best as possible given the data available, which of those events actually occur within our event radius.

Here, our feed position Pipe, [Nassau County Walking Meetup Events], invokes the Pipe `getRichMeetupLocalMeetings` with relevant parameters as follows:

zip	11590
event radius	19.5 <sup>117</sup>
group radius	30
topic list	walkers,hiking
group URL list	<a href="http://adventurers.meetup.com/36/">http://adventurers.meetup.com/36/</a>

For each item returned, the [Proper Case] utility function Pipe is invoked, first with the event title, then with the name of the Meetup group holding the event.<sup>118</sup>

For those events held by private Meetup groups, all but the first item is dropped, and the item retained is given a description that advises the reader of the private status of the group and that they may apply to join the group for more information. For those events held by non-private Meetup groups, the format of the description is modified slightly. The aggregated feed is then output.

### [recurMonthlyWalks]

This Pipe generates a feed consisting of items over the next three months for the monthly general meetings of the Nassau Hiking and Outdoors Club<sup>119</sup> and monthly committee meetings of the Queens Committee of Transportation Alternatives<sup>120</sup>. Still a Phase 2 prototype, this Pipe is due to be reimplemented with the [getMonthOrdinalDayOfWeek] Pipe (described below).

### [getMonthDayOfWeeks]

This Pipe accepts a name of a day of the week, month name, and four-digit year as parameters. It iterates through the numbers 1 through 31, constructing a UTC<sup>121</sup> date string for each day of the given month and year, passes this string to a Date Builder module, and passes that result to a Date Formatter

<sup>117</sup> Our event radius has been increased from 19 miles (the value used in Phase 2), to 19.5 miles, in order to include events held at the boardwalk at Sunken Meadow State Park, in the Town of Smithtown, Suffolk County. This radius also lets some events in upper Manhattan to slip in. Future development will include the creation of Queens and Suffolk specific walking calendars, each of which will focus on narrower radii.

<sup>118</sup> Some Meetup organizers have chosen to identify their group and/or events under their group in all capital letters. As this is the Internet equivalent of shouting, the decision has been made to render these names in proper case.

<sup>119</sup> A non-profit membership organization that conducts hiking, biking and other outdoor activities on and off Long Island, conservation activities and singles events.

<sup>120</sup> Founded in 1973, Transportation Alternatives has a mission is to reclaim New York City's streets from the automobile, and to advocate for bicycling, walking and public transit as the best transportation alternatives. The Queens Committee of Transportation Alternatives was founded in 2007 in response to dramatic changes on the streets of Queens. The goal of this volunteer effort is to seize on the recent surge in bicycling, development and new park space to improve bicycling, walking and public transit conditions in Queens.

<sup>121</sup> *Coordinated Universal Time*. UTC is an acronym that intentionally fails to track to the initials of its referent. (It likewise is mismatched to *Temps Universel Coordonné*, the same phrase in French. The compromise ordering of characters panders to neither culture.) All other time zones are defined as offsets to UTC, and Yahoo! Pipes reduces all datetime values to this time zone to provide a single standard for comparisons.

module<sup>122</sup>, generating a string in the form “Dayofweek, Month Day, Year”. Those items that do not match our day of the week parameter are dropped, and the remaining items output.

### **[getMonthOrdinalDayOfWeek]**

This Pipe accepts a name of a day of the week, month name, four-digit year, and ordinal value as parameters, and invokes the [getMonthDayOfWeeks] Pipe with name of day of week, month name, and four-digit year. All but the item corresponding to the ordinal value are dropped, and the remaining item is output.

### **[testInFloatingDateRange]**

The testInDateRange utility function Pipe accepts a beginning datetime value, ending datetime value, and floating datetime value to test as parameters. The datetime value to test is compared to the beginning datetime value and ending datetime value, and one of three results are output: “before”, “between” or “after”.

This [testInFloatingDateRange] utility function Pipe accepts a beginning floating date and time, ending floating date and time, and floating date and time to test (each expressed as a string without a time zone specified) as parameters. Each string is passed to a Date Builder module to generate a UTC datetime value, and these values are passed to the testInDateRange utility function Pipe. The result is output.

### **[floatToUSAwDST]**

This utility function Pipe accepts a three-letter code for a United States standard time zone and floating date and time (expressed as a string without a time zone specified) as parameters. The date-and-time string is passed to a Date Builder module to obtain a UTC datetime value, and the year of that datetime value is then passed to the [getMonthOrdinalDayOfWeek] module, first with the parameters (2, Sunday, March) and then again with the parameters (1, Sunday, November)<sup>123</sup>. The results of these two operations are used to assemble floating date and time strings representing 2:00 a.m. on both days, and these are passed, together with our floating date and time string parameter, to the utility function Pipe [testInFloatingDateRange]. If the result is “between”, the standard time zone code is changed to a daylight savings time zone code, otherwise the standard time zone code is retained. A string, consisting of our floating date and time string parameter and our time zone, is assembled, and then output.

### **[recurWeeklyWalks]**

This Pipe passes the strings “today” and “next month” each to a Date Builder module, and in turn passes the resulting datetime values to a Date Formatter module to get the name of the current and following month. These values, in their own turn, are passed to the getMonthDayOfWeeks Pipe,

---

<sup>122</sup> The Date Formatter module of Yahoo! Pipes, as its name suggests, accepts a datetime attribute and

<sup>123</sup> Reflecting the second Sunday in March and the first Sunday and November, the dates on which daylight savings time begins and ends in the United States, under the U.S. Energy Policy Act of 2005.

together with the day of week “Saturday” and the respective year of the given month. The items in the resulting feed are then filtered, dropping those dates that are in the past, and then truncated, keeping only the first three future dates. For each of the remaining dates, a floating date and time string is assembled with a time of 8:00 a.m., which is passed, along with a time zone of “EST”, to the [floatToUSAwDST] utility function Pipe.

For each of the resulting date and time strings, an item is assembled describing an event held at that date and time by the Long Island Walkers Club<sup>124</sup>, which items are then output.

### [oldWestburyWalks]

This Pipe generates a feed consisting of items for each of the walking tours scheduled at Old Westbury Gardens<sup>125</sup> on their current season calendar. Still a Phase 2 prototype, this Pipe is due to be reimplemented through an external events submission database (discussed below), pending outreach to Old Westbury Gardens, and local venues like it, to develop its presence on Meetup.com and/or its events calendar in a developer-accessible format (also discussed below).

### [muttontownWalks]

This Pipe generates a feed consisting of items for each of the nature hikes scheduled at Muttontown Preserve<sup>126</sup> on their current season calendar. Still a Phase 2 prototype, this Pipe is due to be reimplemented through an external events submission database, pending outreach to Muttontown Preserve, and local venues like it, to develop its presence on Meetup.com and/or its events calendar in a developer-accessible format.

### [walks not on meetup]

This Pipe aggregates the feeds from the [recurMonthlyWalks], [recurWeeklyWalks], [oldWestburyWalks], and [muttontownWalks] Pipes, performing a series of transformation on the items of each. Still a Phase 2 prototype, this Pipe is due for reimplementation through a combination of an external submission database and refactoring of some functionality to support the first two of the foregoing subpipes.

### [discoverWalks]

This Pipe fetches search results (for local walking and hiking events) from the Long Island Convention Bureau and Sports Commission's Web site<sup>127</sup>, parses this data to determine event dates and locations,

124 The Long Island Walkers Club, originally founded as the New York Walkers Club in 1979 by Coach Jake Jacobson, promotes the health benefits of walking. The group welcomes any adult who would like to walk with others and enjoy the benefits of walking.

125 Old Westbury Gardens, listed on the National Register of Historic Places, welcomes visitors of all ages for guided tours of Westbury House and Gardens, in-depth tours of the formal gardens, and a wide range of other events, including and Master Gardener- and Educator-led Talks & Tours of topics relating to horticulture, art, history, design and architecture.

126 Muttontown Preserve, maintained by the Nassau County Department of Parks, Recreation, and Museums, has been named the "Best Nature Walk of Long Island". Nature walks are hosted at Muttontown Preserve throughout the year.

127 As of August 2, 2008, this Web site is returning errors on attempting to view individual event pages turned up by the search engine, such that our Pipe is unable to obtain sufficient data to produce a feed of these events. This has occurred

and outputs the results. Still a Phase 2 prototype, this Pipe is due for extensive refactoring, pending outreach to the Long Island Convention Bureau and Sports Commission, and other providers of local and community event calendars, to make its calendar available in a developer-accessible format.

### [NassauWalks Calendar]

This Pipe invokes the [discoverWalks] Pipe, and for each item returned: (i) invokes the utility function Pipe [callDistanceJsonUrl], passing it the latitude and longitude of the event, the ZIP Code 11590, the URL of a distance calculation service, and the JSON path for the results of said service; and (ii) filters out those events further than 19.5 miles away.<sup>128</sup> The remaining results are then aggregated with the output of both the [Nassau County Walking Meetup Events] Pipe and the [walks not on meetup] Pipe.

For each item in the aggregated feed, the publication date of the item is set to the date and time of the event<sup>129</sup>, the items are sorted chronologically, and output.

### [Where We're Walking (Text)]

This Pipe invokes the [NassauWalks Calendar] Pipe, drops items more than a month in the future, truncates the feed if it is still longer than 30 items, and outputs the results.

### rss.cgi

This CGI script retrieves the output of the [Where We're Walking (Text)] Pipe (by way of the <tinyurl.com> service), formats each item for incorporation on the nassauwalks.org home page, and returns the result. Unlike the CGI scripts for other features discussed herein, `rss.cgi` only caches the feed for intervals of 30 minutes<sup>130</sup>. This is a compromise to contend with two issues:

First, the Where We're Walking feature (discussed below) is implemented using a Yahoo! Pipes badge, which draws on the pipeline that feeds our [Where We're Walking (Text)] Pipe, without the benefit of the extended caching intervals used by our other, CGI, based features. Thus, there would be no benefit in caching for an extended period—the Pipe gets invoked more frequently in any case.

Second, as discussed at the beginning of the discussion of this feature, the complexity and depth of the pipeline serving this feature is such that timeouts and other transient errors are frequent, and any such corruptions of the feed would inevitably be cached by `rss.cgi`, as would an incomplete feed resulting from the occasional unavailability of the Meetup API service. These issues tend to be resolved in under half an hour. Thus, we cache on a smaller interval so that incomplete or error-laden instances of a feed are not retained for an overly long time.

---

at least once before in this author's experience, and will hopefully be corrected in the near future.

128 The [discoverWalks] Pipe retrieves data by region, as per the functionality of the underlying search engine. The distance filter used here will be incorporated into the [discoverWalks] Pipe when it is refactored.

129 As we are ultimately rendering this feed in RSS format, we are restricted to the data elements native to that format.

RSS is designed for news, blogs, and other content organized by date of publication, where here we are dealing with calendars organized by the future dates on which such events are due to occur. We appropriate the publication date data element to do the work of conveying our calendar dates.

130 As refreshing the cache is a *post hoc* operation, this represents the minimal duration the cache may be maintained.

## **Where We're Walking**

```
- | [DateZoneReformatter]]
```

```
[Where We're Walking (Text)]
  | [Where We're Walking] ([DateZoneReforatter])
```

```
[Where We're Walking] | <pipes.yahoo.com> | <nassauwalks.org>
```

This pipeline extends the functionality of our Local Calendar feature, by displaying the events as markers on a map, such that one can easily see where in the area events are scheduled to occur in the near future. This map is implemented by a Javascript widget provided by <pipes.yahoo.com> and embedded at the top of the main content area of the nassauwalks.org home page.

This feature, unfortunately, has a habit of timing out, such that no results are returned, and a large blank space is left on the nassauwalks.org home page. A caching mechanism to retain good result sets is due for development. (This will also have the advantage of reducing how frequently the underlying pipeline is invoked.)

### **[DateZoneReformatter]**

This Pipe accepts a date, time, and timezone formatted as a string and a pattern for formatting a date as parameters, substitutes “UTC” for the timezone, and passes the result to a Date Builder module. This datetime value is then passed to a Date Formatter module, using the pattern parameter provided.

Meanwhile, the timezone of the original string parameter is extracted. If the format pattern included a timezone, the UTC timezone in the Date Formatter result is replaced with the timezone from the original string parameter. The result is then output.

### **[Where We're Walking]**

This Pipe invokes the [Where We're Walking (Text)] Pipe, and for each item performs the following operations: (i) invokes the [DateZoneReformatter] utility function Pipe, passing it the date and time of the event; (ii) performs additional regex operations on the output of [DateZoneReformatter], in order to obtain a minimalist, floating date-time string; (iii) replaces the event description with the group photo (if any), our minimalist date-time string, and the group name; (iv) filters out phantom items that identify events of the Nassau County Walking Meetup Group as private<sup>131</sup>; and (v) for each private group event, the string “(private group)” is appended to the description. The results are sorted chronologically and output.

---

<sup>131</sup> This was implemented to address a specific issue, which may have since been resolved elsewhere.

## Future Development

There is still a lot to be done. What follows is a quick overview of work pending. To expedite the release of this document, we will spare the extensive discussion and simply resort to bulleted lists.

### **Technical**

- development of Meetup API and Yahoo! Pipes user groups to share ideas and solutions
- JSON-caching service component to compensate for timeouts within pipelines
- caching component in badge pipeline
- reimplementing of remaining Phase 2 prototypes
- reimplementing of topic-oriented Pipes to use RSS topic feeds rather than API methods
- elimination of components that currently compensate for limitations in Meetup API (once those limitations have been addressed)
- conversion of current SSI/CGI-based implementations to AJAX
- WAP-compliant messaging application, to allow organizers to contact members who have RSVP'd for events from handheld wireless devices
- proposal for SMS based reservation interface to Meetup.com database
- proposal for Interactive Voice Response/Automated Voice Response Callout reservation interface to Meetup.com database

### **Functional**

- synchronization of Meetup group calendars across other calendaring platforms
- outreach with providers of local and community calendars to include the calendar feeds of local Meetup groups
- synchronization of topical notes of non-Meetup events on Meetup calendars
- wizard to simplify the task of recasting calendar notes as meetings
- annotation feature for calendar notes, to allow members to share their thoughts about upcoming non-Meetup events of interest
- roll-over sign-up and RSVP features (once a suitable user-auth mechanism is implemented as part of the Meetup API)
- implementation of full *looking glass* functionality (full discussion of this will have to wait)
- syndication of Meetup group events to offline publications (once an IVR/AVRC system is in place)

## ***Content***

- inclusion of local walking/hiking related events on the calendar of the New York State Office of Parks, Recreation and Historic Preservation
- external events submission database (for one-off events and groups with short seasonal calendars)
- population of said database with events from numerous local groups currently unrepresented, and outreach to said groups to participate in maintaining the database
- outreach to venues to either develop Meetup groups or make calendars available in developer-accessible formats
- outreach to providers of local and community calendars to make their calendars available in developer-accessible formats

## ***Geographic***

- walking-oriented Web pages serving Queens and Suffolk County (western and eastern)
- personalized member e-mail calendar/newsletter
- distance-sorted Meetup alerts utility (once an alerts.json method is available, and assuming Meetup doesn't finally add this functionality to the existing alerts feature)
- development of county and local Meetup activity dashboards, to help organizers in regions of comparatively sparse Meetup penetration target their group development efforts
- wizard for generating promotional flyers that highlight specific Meetup group events in the context of other Meetup events going on nearby

## ***Topics***

- development of Meetup API user groups to foster the development of other topic-oriented applications
- wizard for generating promotional flyers that highlight a specific Meetup group in the context of other nearby Meetup groups about the same topic
- re-purposing of components developed for nassauwalks.org to create a Long Island free summer concerts Web site, aggregating the event calendars of area parks, towns, villages, libraries, and other venues